SKRIPSI

VISUALISASI KURIKULUM 2018 DENGAN VIS.JS DAN ELECTRON

Joshua Delavo Setiadi

NPM: 2017730028

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022

UNDERGRADUATE THESIS

VISUALIZATION CURRICULUM 2018 WITH VIS.JS AND ELECTRON

Joshua Delavo Setiadi

NPM: 2017730028

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022

LEMBAR PENGESAHAN

VISUALISASI KURIKULUM 2018 DENGAN VIS.JS DAN ELECTRON

Joshua Delavo Setiadi

NPM: 2017730028

Bandung, 09 Januari 2022

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, Nugroho, M.Comp.	«pembimbing pendamping/2»

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

VISUALISASI KURIKULUM 2018 DENGAN VIS.JS DAN ELECTRON

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 09 Januari 2022

Joshua Delavo Setiadi
NPM: 2017730028

ABSTRAK

Setiap mahasiswa Universitas Katolik Parahyangan jurusan informatika perlu melakukan pengisian formulir rencana studi (FRS) untuk pengambilan matakuliah di semester berikutnya. Untuk dapat mengetahui mata kuliah apa saja yang dapat diambil pada semester berikutnya, mahasiswa harus melihat pada pohon kurikulum yang terdapat pada buku petunjuk pelaksanaan kegiatan akademik (juklak). Namun, pohon kurikulum tersebut memiliki kekurangan dimana tidak terdapat mata kuliah pilihan dan sulit untuk melihat prasyarat pada setiap mata kuliah, karena penggambaran garis prasyarat pada pohon kurikulum memiliki warna yang sama danjuga saling bertumpuk.

Aplikasi VisKur akan mengambil data dari API milik FTIS Unpar kemudian memvisualisasikannya dalam bentuk *Network* dan *Timeline* menggunakan *framework Electron* dengan *library Vis.js*. Viskur dibuat menggunakan JavaScript dan HTML.

Setelah dilakukan pengujian terhadap aplikasi VisKur dengan melakukan survei kepada beberapa mahasiswa aktif Unpar jurusan informatika, didapatkan hasil bahwa aplikasi VisKur dapat menyajikan kurikulum 2018 dengan lebih baik dibanding dengan pohon kurikulum yang terdapat pada juklak. Namun, untuk saat ini VisKur hanya dapat berjalan pada sistem operasi Windows saja.

Kata-kata kunci: VisKur, Electron, Vis.js, Javascript, HTML, juklak, informatika Unpar

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: VisKur, Electron, Vis.js, Javascript, HTML, juklak, informatics Unpar

Untuk orang tua yang telah membiayai pendidikan saya sampai saat ini

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena telah memberikan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi yang berujul "Visualisasi Kurikulum 2018 dengan Vis.js dan Electron" dengan baik. Skripsi ini disusun dengan tujuan untuk memenuhi salah satu prasyarat kelulusan di Jurusan Teknik Informatika , Fakultas Teknologi Informasi dan Sains, Universitas Katolik Parahyangan. Dalam pengerjaannya, penulis dibantu oleh beberapa pihak, oleh karena itu, penulis ingin mengucapkan terima kasih yang sebesar besarnya kepada:

- Allah Bapa, Allah Anak, dan Allah Roh Kudus karena telah memberkati penulis dalam mengerjakan skripsi ini sehinga penulis bisa mendapatkan hasil yang baik dan dinyatakan lulus.
- Keluarga penulis yang telah memberikan dukungan penuh dalam hal material dan mental.
- Bapak Pascal Alfadian Nugroho, S.Kom., M.Comp. selaku pembimbing yang senantiasa sabar memberikan kritik dan saran untuk membantu pembuatan skripsi ini pada setiap sesi bimbingan. Terima kasih karena telah memberikan kepercayaan kepada penulis untuk dapat maju di sidang akhir.
- Ibu Vania Natalie, S.Kom., M.T. selaku dosen penguji yang telah memberikan kritik dan saran kepada penulis.
- Teman - teman informatika angkatan 2017 yang telah memberikan banyak dukungan kepada penulis dalam mengerjakan skripsi ini.

Semoga semua pihak yang telah membantu diberikan berkat oleh Tuhan dan dapat berhasil serta sukses dengan semua kegiatannya. Semoga skripsi ini juga dapat bermanfaat bagi orang yang membacanya. Penulis juga memohon maaf apabila ada kesalahan dan kekurangan dalam penulisan skripsi ini.

Bandung, Januari 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Electron adalah salah satu kerangka kerja terbaru yang memungkinkan pengembang membuat aplikasi *native desktop* dengan teknologi web populer : JavaScript, HTML5, dan CSS. Dengan *Electron*, pengembang web dapat menggunakan keterampilan yang mereka miliki untuk membangun aplikasi yang memiliki banyak kemampuan seperti aplikasi *native desktop*. *Elektron* telah menjadi sangat populer sejak dirilis dan digunakan oleh perusahaan, seperti : *Microsoft*, *Facebook*, *Slack*, dan *Docker*. Aplikasi ini dapat dikemas untuk dapat berjalan langsung di *macOS*, *Windows*, dan *Linux*. Bisa juga didistribusikan melalui *Mac App Store* atau *Microsoft Store*.

Vis.js adalah sebuah *library* visualisasi berbasis *browser* yang bersifat dinamis. *Library* ini dirancang agar mudah digunakan untuk menangani data dinamis (berubah secara *realtime*) dalam jumlah yang besar dan memungkinkan untuk memanipulasi serta berinteraksi dengan data tersebut. *Library* ini terdiri dari komponen - komponen, seperti *DataSet*, *Timeline*, *Network* (*tree*), *Graph2d*, dan *Graph3d*. *DataSet* berfungsi untuk mengelola data yang tidak terstruktur. Terdapat fitur *add*, *update*, dan *remove* data di dalamnya. *Timeline* berfungsi untuk menampilkan data dalam bentuk *timeline* yang dapat disesuaikan dengan *item* dan rentangnya. *Network* (*tree*) berfungsi untuk menampilkan data dalam bentuk jaringan yang dinamis, dapat diatur secara otomatis, dan dapat disesuaikan. *Graph2d* berfungsi untuk menampilkan data dalam bentuk grafik dan diagram batang pada *timeline* yang interaktif sesuai yang diinginkan. *Graph3d* berfungsi untuk menampilkan data dalam bentuk grafik 3d dengan animasi yang interaktif.

GitHub adalah sebuah aplikasi berbasis *website* dengan *Version Control System* (VCS) yang menyediakan layanan untuk menyimpan *repository* dengan gratis. VCS adalah sebuah infrastruktur yang dapat mendukung pengembangan *software* secara kolaboratif. Setiap anggota yang berada di dalam sebuah tim pengembangan *software* dapat menulis kode programnya masing - masing kemudian digabungkan ke server yang sudah memiliki VCS yang digunakan. *Respository* merupakan tempat yang dapat digunakan untuk menyimpan berbagai file berupa *source code*. Aplikasi ini termasuk sangat populer dan banyak digunakan termasuk oleh perusahaan - perusahaan besar, seperti : *Facebook*, *Google*, dan *Twitter*.

Saat mahasiswa akan melakukan pengisian formulir rencana studi (FRS), seringkali mereka kesulitan untuk melihat kurikulum tahun ajaran yang berlaku. Dikarenakan setiap kurikulum memiliki aturan yang berbeda dalam pengambilan matakuliah ataupun matakuliah yang disediakan ditambah dengan tidak adanya mata kuliah pilihan untuk setiap semesternya, maka mahasiswa kadang bingung untuk memilih matakuliah apa yang akan diambil di semester berikutnya.

Maka dari itu, pada skripsi ini akan dibuat sebuah aplikasi visualisasi kurikulum 2018 berbasis *Electron* dengan menggunakan *library Vis.js*. Aplikasi ini akan saya namakan VisKur. Dengan aplikasi ini diharapkan mahasiswa dapat lebih mudah untuk melihat kurikulum yang ada sehingga mempermudah mereka untuk memilih matakuliah apa yang akan diambil di semester berikutnya, kemudian dengan penggunaan *framework cross platform Electron* ini diharapkan semua mahasiswa pengguna *macOs*, *Windows*, dan *Linux* dapat mengaksesnya dengan mudah.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada skripsi ini adalah :

1. Visualisasi bentuk apa yang paling cocok untuk memvisualisasikan kurikulum 2018?
2. Bagaimana mengimplementasikan visualisasi tersebut dalam *Vis.js*?
3. Bagaimana cara membaca kurikulum 2018 FTIS UNPAR dari *github*?

1.3 Tujuan

Tujuan yang akan dicapai dari penulisan skripsi ini adalah :

1. Memahami bentuk apa yang paling cocok untuk memvisualisasikan kurikulum 2018.
2. Memahami cara mengimplementasikan visualisasi tersebut dalam *Vis.js*.
3. Memahami cara membaca data kurikulum 2018 FTIS UNPAR dari *github*.

1.4 Batasan Masalah

Batasan-batasan masalah yang ditetapkan adalah sebagai berikut :

1. Kurikulum 2018 yang digunakan hanya milik program studi teknik informatika.
2. Karena keterbatasan mesin yang dimiliki, maka hanya dibuat dalam versi Windows saja
3. Perangkat lunak ini hanya akan menggunakan bahasa *HTML* dan *JavaScript*

1.5 Metodologi

Bagian - bagian pengerjaan skripsi ini adalah :

1. Melakukan studi tentang *framework Electron* dan *Library Vis.js*.
2. Mempelajari cara membuat aplikasi berbasis *Electron*.
3. Mempelajari cara memvisualisasikan data dalam bentuk *tree* dan *timeline* dengan *Vis.js*.
4. Mempelajari data kurikulum 2018 di github beserta cara pengambilan datanya.
5. Merancang aplikasi berbasis *Electron*.
6. Merancang visualisasi kurikulum 2018 dalam bentuk *tree*.
7. Merancang visualisasi kurikulum 2018 dalam bentuk *timeline*.
8. Mendesain antarmuka aplikasi.
9. Melakukan pengujian dan ekperimen.
10. Membuat dokumen skripsi.

1.6 Sistematika Pembahasan

Skripsi ini terdiri dari enam bab, yaitu pendahuluan, landasan teori, analisis, perancangan, implementasi, dan kesimpulan dan saran.

Bab I membahas latar belakang dibuatnya skripsi, rumusan masalah yang terdapat pada skripsi, tujuan skripsi ini dibuat, batasan masalah agar skripsi yang dibuat tidak terlalu luas, dan metodologi yang berisi langkah - langkah pengerjaan skripsi agar berjalan sistematis.

Bab II berisi teori - teori yang berfungsi sebagai referensi dalam pembuatan skripsi dan membantu dalam menyelesaikan masalah pada skripsi.

Bab III berisi analisis terhadap perangkat lunak yang telah dibuat.

Bab IV berisi perancangan perangkat lunak menggunakan aplikasi *Electrorn* dan *library Vis.js*.

Bab V berisi implementasi perangkat lunak yang berlandaskan teori - teori yang telah dipelajari.

Bab VI berisi kesimpulan skripsi yang telah dibuat dan juga saran yang ditujukan untuk skripsi berikutnya.

BAB 2

LANDASAN TEORI

Bab Landasan Teori ini berisi teori-teori yang menjadi dasar penelitian ini, meliputi kurikulum 2018, *Electron*, *Vis.js*, *FTIS Open Data*, *dan JavaScript*.

2.1 Kurikulum 2018

Kurikulum didefinisikan sebagai seperangkat rencana dan pengaturan mengenai capaian pembelajaran lulusan, bahan kajian, proses, dan penilaian yang digunakan sebagai pedoman penyelenggaraan program studi menjadi sarana utama untuk mencapai tujuan tersebut. [1]

Penyusunan Kurikulum 2018 berpegang pada prinsip bahwa kurikulum yang baik adalah kurikulum yang tidak hanya kokoh, secara teoretis konseptual dapat dipertanggungjawabkan, namun juga secara praktis dapat dilaksanakan. Selain itu kurikulum juga harus cukup fleksibel agar dapat mengakomodasi perubahan-perubahan, namun tanpa kehilangan ciri atau kekhasan dari program studi.

Dalam penyusunan Kurikulum 2018 Program Studi Teknik Informatika secara khusus juga memperhatikan Kerangka Kualifikasi Nasional Indonesia (KKNI) yang tertuang dalam Peraturan Presiden no 8 tahun 2012. KKNI merupakan pernyataan kualitas SDM Indonesia, di mana tolok ukur kualifikasinya ditetapkan berdasarkan capaian pembelajaran (learning outcomes) yang dimilikinya. Penyusunan kurikulum mengikuti tahapan perancangan kurikulum yang disarankan oleh Kemenristekdikti yang diberikan pada Gambar 2.1. Tahapan penyusunan kurikulum 2018 meliputi kegiatan sebagai berikut:

1. Melakukan evaluasi diri
2. Merumuskan profil lulusan dengan pelacakan lulusan
3. Menentukan capaian pembelajaran
4. Menentukan bahan kajian
5. Menyusun matriks pembelajaran dan bahan kajian
6. Membentuk mata kuliah
7. Menyusun struktur kurikulum dan menentukan metode pembelajaran

Gambar 2.1: Tahapan penyusunan kurikulum

2.1.1 Kodifikasi

Kodifikasi tiap mata kuliah dibuat berdasarkan Peraturan Rektor UNPAR No. III/PRT/2017-03/46 tentang Standar Penyusunan Kurikulum Program Studi di Lingkungan UNPAR. Kode ini terdiri atas 11 dijit, dengan rincian berikut:

- 3 digit - kode khas Program Studi: AIF
- 2 digit - tahun diberlakukannya kurikulum (2 digit terakhir): 18
- 1 digit - urutan tahun pengajaran
- 1 digit - nomor urut KBI pengampu mata kuliah
- 2 digit - nomor urut mata kuliah per semester, dengan angka pada dijit terakhir sebagai penentu semester; ganjil atau genap
- 2 digit - jumlah sks mata kuliah

Informasi lengkap terkait kodifikasi ini diberikan di Gambar 2.2.

Penyelenggara	Universitas	Prodi
Kode khas prodi	MKU	AIF
Tahun berlaku kurikulum	18	18
Urutan tahun pengajaran	0	1: tahun pertama 2: tahun kedua 3: tahun ketiga 4: tahun keempat
Nomor urut KBI pengampu	**	0: Prodi 1: Teori Komputasi 2: Sistem Terdistribusi 3: Sistem Informasi
Nomor urut mata kuliah	**	Urutan mata kuliah per semester, dengan angka pada dijit terakhir sebagai penentu semester; ganjil atau genap
Jumlah sks	**	Jumlah sks

**Kode mata kuliah MKU ditentukan oleh universitas

Gambar 2.2: Kodifikasi mata kuliah

2.1.2 Bobot Pemrograman

Berdasarkan hasil evaluasi Kurikulum 2013, salah satu masalah yang ditemukan adalah bahwa mahasiswa masih sulit menguasai materi kuliah di jalur pemrograman, yang merupakan kuliah inti dari Prodi Teknik Informatika UNPAR. Selain karena memang logika pemrograman tidak mudah

untuk dipahami, kurangnya pengalaman mahasiswa dalam membangun program komputer juga menjadi penyebab munculnya permasalahan ini. [1]

Selain memperbaiki struktur kuliah jalur pemrograman, dan perbaikan materi perkuliahan, cara lain yang digunakan untuk mendukung kemampuan pemrograman mahasiswa adalah dengan menempatkan bobot pemrograman di kuliah-kuliah yang cocok. Bobot pemrograman ini menentukan di kuliah mana saja mahasiswa harus membangun program komputer, dan seberapa besar skala program komputer yang dibuat. Bagian pembangunan program komputer misalnya dapat diletakkan pada saat praktikum, atau dijadikan bagian dari tugas kuliah.

Besar bobot pemrograman dalam kurikulum ini adalah 0.25, 0.5, 0.75, dan 1. Penjelasan terkait masing - masing bobot ini diberikan pada Gambar 2.3.

Bobot	Deskripsi
0.25	• Minimal 1 tugas berbentuk pembangunan program komputer • Kuliah tidak berpraktikum
0.5	• Minimal setengah dari tugas yang diberikan berbentuk pembangunan program komputer • Kuliah tidak berpraktikum atau yang berfokus pada analisis
0.75	• Di luar tugas praktikum, ada tugas kuliah berupa pembangunan program komputer • Kuliah berpraktikum atau merupakan kuliah skripsi
1	Kuliah berpraktikum dengan capaian pembelajaran adalah keahlian pemrograman atau merupakan kuliah proyek

Gambar 2.3: Rincian Bobot Pemrograman

2.1.3 Prasyarat Mata Kuliah

Di Prodi Teknik Informatika terdapat 2 jenis prasyarat, yaitu prasyarat lulus dan prasyarat tempuh. Prasyarat lulus artinya seorang mahasiswa harus lulus mata kuliah prasyarat (nilai minimum D), baru dapat mengambil suatu mata kuliah, sedangkan prasyarat tempuh artinya seorang mahasiswa harus pernah menempuh mata kuliah prasyarat, sebelum dapat mengambil suatu mata kuliah. Rincian prasyarat mata kuliah wajib diberikan pada Gambar 2.4, Gambar 2.5, Gambar 2.6, Gambar 2.7, Gambar 2.8, Gambar 2.9, Gambar 2.10, dan Gambar 2.11, sedangkan rincian prasayarat mata kuliah pilihan diberikan pada Gambar 2.12, Gambar 2.13, Gambar 2.14, dan Gambar 2.15.

No	Kode	Mata Kuliah	Mata Kuliah Prasyarat	
			Tempuh	Lulus
Semester 1				
1	AIF181101-03	Pemodelan untuk Komputasi		
2	AIF181103-04	Matematika Dasar		
3	AIF181105-02	Pengantar Informatika		
4	AIF181107-03	Matematika Diskret		
5	MKU180130-02	Bahasa Indonesia		
6	MKU180110-02	Pendidikan Kewarganegaraan		
7	MKU180120-02	Logika		

Gambar 2.4: Daftar mata kuliah wajib semester satu beserta prasyaratnya

Semester 2				
1	AIF181100-04	Dasar Pemrograman		Mulai angkatan 2018: AIF181101-03
2	AIF181202-04	Arsitektur dan Organisasi Komputer		
3	AIF181104-03	Logika Informatika		
4	AIF181106-03	Matriks dan Ruang Vektor		
5	MKU180240-02	Etika		
6	MKU180360-02	Estetika		

Gambar 2.5: Daftar mata kuliah wajib semester dua beserta prasyaratnya

Semester 3				
1	AIF182101-03	Algoritma dan Struktur Data		AIF181100-04
2	AIF182103-04	Struktur Diskret	AIF181107-03	
3	AIF182105-02	Pemrograman Berorientasi Objek		AIF181100-04
4	AIF182007-02	Teknik Presentasi		
5	AIF182109-03	Statistika untuk Komputasi		
6	MKU180370-02 / MKU180380-02	Agama Katolik/Fenomenologi Agama		
7	MKU180250-02	Pancasila		

Gambar 2.6: Daftar mata kuliah wajib semester tiga beserta prasyaratnya

Semester 4				
1	AIF182100-04	Analisis dan Desain Perangkat Lunak		AIF182105-02
2	AIF182302-04	Manajemen Informasi dan Basis Data	AIF182101-03	
3	AIF182204-03	Pemrograman Berbasis Web	AIF182302-04 (bersamaan atau sudah tempuh)	
4	AIF182106-03	Desain dan Analisis Algoritma	AIF182103-04	AIF182101-03
5	AIF182308-03	Pengantar Sistem Informasi	AIF182302-04 (bersamaan atau sudah tempuh)	AIF181105-02
6	AIF182210-02	Pengantar Jaringan Komputer		

Gambar 2.7: Daftar mata kuliah wajib semester empat beserta prasyaratnya

Semester 5				
1	AIF183201-03	Sistem Operasi	AIF182101-03	
2	AIF183303-03	Rekayasa Perangkat Lunak	AIF182100-04	
3	AIF183305-02	Manajemen Proyek	AIF183303-03 (bersamaan atau sudah tempuh)	
4	AIF183107-03	Pengantar Sistem Cerdas	AIF182106-03 AIF181104-03	
5	AIF183209-03	Pemrograman pada Perangkat Bergerak	AIF182210-02 AIF182100-04	
6	AIF183111-03	Interaksi Manusia Komputer		

Gambar 2.8: Daftar mata kuliah wajib semester lima beserta prasyaratnya

Semester 6				
1	AIF183300-02	Teknologi Basis Data		AIF182302-04
2	AIF183002-02	Penulisan Ilmiah		
3	AIF183204-02	Jaringan Komputer	AIF182210-02	
4	AIF183106-06	Proyek Informatika	AIF183305-02	
	AIF183308-03	Proyek Sistem Informasi 1	AIF183305-02	AIF182308-03
	AIF183310-03	Proyek Data Science 1	AIF183305-02	AIF182109-03

Gambar 2.9: Daftar mata kuliah wajib semester enam beserta prasyaratnya

Semester 7				
1	AIF184001-03	Skripsi 1		AIF183002-02 Sudah lulus 108 sks Mulai angkatan 2017: AIF183002-02 AIF182007-02
				Sudah lulus 108 sks
2	AIF184303-03	Proyek Sistem Informasi 2		AIF183308-03
	AIF184305-03	Proyek Data Science 2		AIF183310-03
3	AIF184005-02	Komputer dan Masyarakat		

Gambar 2.10: Daftar mata kuliah wajib semester tujuh beserta prasyaratnya

Semester 8				
1	AIF184000-02	Etika Profesi		
2	AIF184002-05	Skripsi 2		AIF184001-03 Jika diambil bersamaan dengan AIF184001-03 Prasyarat: lulus AIF183002-02 AIF182007-02 dan lulus 124 sks
3	AIF184004-08	Tugas Akhir		AIF183002-02 Sudah lulus 124 sks Mulai angkatan 2017: AIF183002-02 AIF182007-02 Sudah lulus 124 sks

Gambar 2.11: Daftar mata kuliah wajib semester delapan beserta prasyaratnya

No	Kode	Mata Kuliah	Prasyarat	
			Tempuh	Lulus
1	AIF182111-03	Pemrograman Kompetitif 1		AIF182101-03 (minimum C)
2	AIF182001-03	Penelitian 1		AIF181100-04
3	AIF182301-03	Pengantar Data Science		
4	AIF182112-03	Pemrograman Kompetitif 2		AIF182111-03 (minimum B)
5	AIF182102-03	Statistika dengan R		AIF182109-03
6	AIF182002-03	Penelitian 2		AIF181100-04
7	AIF183013-02	Kerja Praktek 1		
8	AIF183015-03	Pendidikan Pengabdian kepada Masyarakat		

Gambar 2.12: Daftar mata kuliah pilihan beserta prasyaratnya (1)

No	Kode	Mata Kuliah	Prasyarat	
			Tempuh	Lulus
9	AIF183117-02	Grafika Komputer	AIF181103-04	AIF182105-02
10	AIF183119-02	Keamanan Informasi		AIF181107-03
11	AIF183121-03	Pemrograman Kompetitif 3		AIF182112-03 (minimum B)
12	AIF183123-02	Topik Khusus Informatika 1		
13	AIF183113-03	Statistika Multivariat dengan R	AIF182102-03	
14	AIF183003-03	Penelitian 3		AIF181100-04
15	AIF183225-03	Sertifikasi Administrasi Jaringan Komputer 1		
16	AIF183229-02	Topik Khusus Sistem Terdistribusi 1		
17	AIF183331-03	Sistem e-Commerce		AIF182308-03
18	AIF183329-03	Basis Data dan Pemrograman SQL untuk Big Data		AIF182302-04 AIF182210-02
19	AIF183335-03	Pengantar Penambangan Data dengan Python	AIF182302-04 (atau tempuh bersama) AIF182109-03 (atau tempuh bersama)	
20	AIF183337-02	Topik Khusus Sistem Informasi 1		
21	AIF183339-02	Sertifikasi Perancangan dan Pemrograman Basis Data dengan Oracle	AIF182302-04	
22	AIF183341-03	Pola Komputasi Big Data	AIF182101-03 AIF182210-02	
23	AIF183143-03	Pemodelan Formal		AIF181104-03
24	AIF183147-02	Sertifikasi Dasar-dasar Java	AIF182105-02	
25	AIF183155-03	Metode Numerik		AIF181103-04 AIF181100-04
26	AIF183203-02	Internet of Things		AIF182210-02
27	AIF183010-03	Kerja Praktek 2		
28	AIF183112-02	Pengujian Perangkat Lunak		AIF183303-03
29	AIF183114-03	Algoritma Kriptografi	AIF183119-02	
30	AIF183116-02	Komputasi Paralel		AIF182101-03

Gambar 2.13: Daftar mata kuliah pilihan beserta prasyaratnya (2)

No	Kode	Mata Kuliah	Prasyarat	
			Tempuh	Lulus
31	AIF183120-03	Pemrograman Permainan Komputer		AIF182101-03 (minimum B)
32	AIF183122-03	Pemodelan Simulasi	AIF182101-03	
33	AIF183128-03	Topik Khusus Informatika 2		
34	AIF183232-03	Pemrograman Berbasis Web Lanjut		AIF182204-03 AIF182302-04
35	AIF183236-03	Sertifikasi Administrasi Jaringan Komputer 2		AIF183225-03
36	AIF183238-03	Topik Khusus Sistem Terdistribusi 2		
37	AIF183240-03	Sertifikasi Cyber Ops		AIF182210-02
38	AIF183342-03	Kewirausahaan Berbasis Teknologi		Sudah lulus 90 sks
39	AIF183346-03	Topik Khusus Sistem Informasi 2		
40	AIF183348-03	Sistem Kecerdasan Bisnis	AIF182302-04	
41	AIF183350-02	IBM Professional Data Science Certificate 1		AIF183335-03
42	AIF184007-04	Kerja Praktek 3		
43	AIF184109-03	Pembelajaran Mesin		AIF183107-03
44	AIF184115-02	Pencarian dan Temu Kembali Informasi		AIF181103-04
45	AIF184119-02	Perancangan Permainan Komputer		AIF182100-04 (minimum B) AIF183120-03 (minumum B)
46	AIF184123-03	Teknologi Mesin Pencari	AIF181106-03	
47	AIF184125-03	Pengolahan Bahasa Alami		AIF183107-03
48	AIF184129-03	Sertifikasi Administrasi Jaringan Komputer 3		AIF183236-03
49	AIF184235-03	Layanan Berbasis Web		AIF182204-03 AIF182302-04 AIF183204-02
50	AIF184339-03	Pengendalian dan Audit Teknologi Informasi	AIF182308-03	
51	AIF184341-03	Penambangan Data		AIF182101-03

Gambar 2.14: Daftar mata kuliah pilihan beserta prasyaratnya (3)

No	Kode	Mata Kuliah	Prasyarat	
			Tempuh	Lulus
52	AIF184337-03	IBM Professional Data Science Certificate 2		AIF183350-02
53	AIF184351-03	Analisis Big Data	AIF183335-03 AIF183341-03 (atau tempuh bersama)	
54	AIF184247-03	Jaringan Komputer Lanjut		AIF183204-02
55	AIF184006-05	Kerja Praktek 4		
56	AIF184106-02	Analisis Data Permainan Komputer		AIF184119-02 (minimum B)
57	AIF184116-02	Sistem Multi Agen	AIF183201-03 AIF183107-03	
58	AIF184222-03	Sertifikasi Administrasi Jaringan Komputer 4		AIF184129-03
59	AIF184334-03	Sistem Informasi Skala Besar		AIF182308-03
60	AIF184338-03	Manajemen Proses Bisnis	AIF182105-02 AIF182204-03	
61	AIF184332-03	Teknologi Big Data dan Cloud Computing	AIF184351-03	
62	AIF184352-03	IBM Professional Data Science Certificate 3		AIF184337-03
63	AIF184330-02	Big Data dan Machine Learning dengan Google Cloud Platform	AIF184351-03 (atau tempuh bersama)	
64	AIF184328-03	Data Science pada Domain Spesifik		AIF183113-03 atau AIF183335-03 atau AIF184351-03

Gambar 2.15: Daftar mata kuliah pilihan beserta prasyaratnya (4)

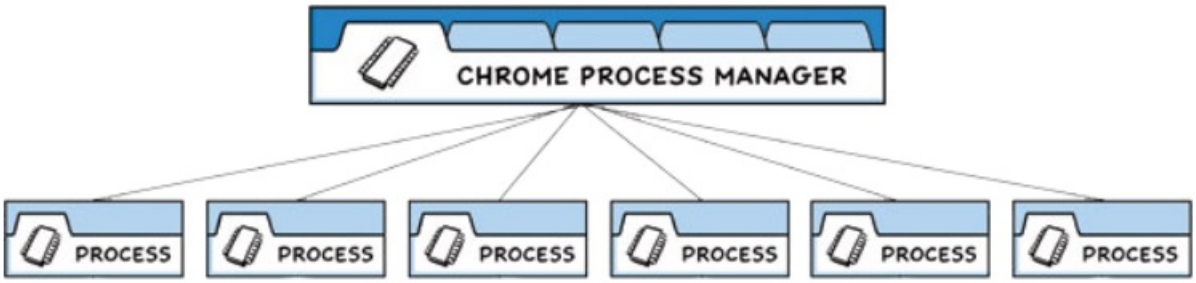
2.2 Electron

Electron adalah sebuah *framework* untuk membangun aplikasi *desktop* menggunakan *JavaScript*, *HTML*, dan *CSS*. Dengan menyematkan *Chromium* dan *Node.js* ke dalam binernya, *Electron* memungkinkan kita untuk mempertahankan satu basis kode *JavaScript* dan membuat aplikasi lintas *platform* yang berfungsi di *Windows*, *macOS*, dan *Linux*, dimana tidak diperlukan pengalaman *native development*. [2]

Electron mewarisi arsitektur *multi-process* dari *Chromium*, yang membuat kerangka kerja secara arsitektur sangat mirip dengan *browser web* moderen. Selanjutnya, akan dijelaskan konsep pengetahuan tentang *Electron*. [3]

Browser web merupakan sebuah aplikasi yang sangat rumit. Selain kemampuan utama mereka untuk menampilkan konten web, mereka memiliki banyak tanggung jawab sekunder, seperti mengelola beberapa jendela atau tab dan memuat ekstensi pihak ketiga. Pada hari - hari sebelumnya, browser biasanya menggunakan satu proses untuk semua fungsi ini. Meskipun pola ini lebih sedikit *overhead* untuk setiap tab yang dibuka, itu juga berarti bahwa satu situs web yang mogok atau macet akan memengaruhi seluruh browser.

Untuk mengatasi masalah tersebut, tim *Chrome* memutuskan bahwa setiap tab akan melakukan *render* untuk setiap prosesnya masing - masing, membatasi bahaya yang dapat ditimbulkan oleh kode berbahaya pada halaman web pada aplikasi secara keseluruhan. Satu proses browser kemudian mengontrol proses ini, serta siklus hidup aplikasi secara keseluruhan. Gambar 2.16 dari *Komik Chrome* memvisualisasikan model ini:



Gambar 2.16: Proses *render Chrome*

Aplikasi *Electron* terstruktur sangat mirip. Sebagai pengembang aplikasi, kita mengontrol dua jenis proses : *main* dan *renderer*. Ini adalah analog dengan browser Chrome dan proses *renderer* yang telah diuraikan di atas. Setiap aplikasi *Electron* memiliki satu proses utama, yang bertindak sebagai titik masuk aplikasi. Proses utama berjalan di lingkungan Node.js, artinya ia memiliki kemampuan untuk membutuhkan modul dan menggunakan semua *Node.js APIs*.

Tujuan dari proses utama adalah untuk membuat dan mengelola jendela aplikasi dengan modul *BrowserWindow*. Setiap contoh dari kelas *BrowserWindow* membuat jendela aplikasi yang memuat halaman web dengan proses *renderer* yang terpisah. Kita dapat berinteraksi dengan konten web ini dari proses utama menggunakan jendela objek webContents. Contohnya dapat dilihat pada Kode 2.1.

Kode 2.1: Contoh kode penggunaan BrowserWindow

```
1 const { BrowserWindow } = require('electron')
2
3 const win = new BrowserWindow({ width:800, height: 1500})
4 win.loadURL('https://github.com')
5
6 const contents = win.webContents
7 console.log(contents)
```

Karena modul *BrowserWindow* adalah *EventEmitter*, kita juga dapat menambahkan *handlers* untuk berbagai aktivitas pengguna (misalnya, meminimalkan atau memaksimalkan jendela). Saat instance *BrowserWindow* dimusnahkan, proses *renderer* yang terkait akan dihentikan.

Proses utama juga mengontrol siklus hidup aplikasi melalui modul aplikasi *Electron*. Modul ini menyediakan serangkaian besar kejadian dan metode yang dapat digunakan untuk menambahkan perilaku aplikasi khusus (misalnya, menutup aplikasi secara terprogram, atau memodifikasi dokumentasi aplikasi). Contohnya, pada Kode 2.2 menggunakan API aplikasi untuk membuat pengalaman aplikasi *window* menjadi lebih nyata.

Kode 2.2: Contoh kode untuk keluar dari aplikasi

```
app.on('window-all-closed', function(){
    if(process.platform !== 'darwin') app.quit()
})
```

Setiap aplikasi *Electron* memunculkan proses penyaji terpisah untuk setiap *BrowserWindow* yang terbuka. Seperti namanya, perender bertanggung jawab untuk merender konten web. Untuk semua maksud dan tujuan, kode yang dijalankan dalam proses perender harus berperilaku sesuai dengan standar web (setidaknya sejauh yang dilakukan Chromium). Oleh karena itu, semua pengguna *interface* dan fungsionalitas aplikasi dalam satu jendela browser harus ditulis dengan alat dan paradigma yang sama dengan yang digunakan di web. Terdapat beberapa spesifikasi web yang harus dipahami, seperti :

- File HTML untuk proses rendering.
- Menambahkan styling UI melalui Cascading Style Sheets (CSS).
- Kode JavaScript yang dapat dieksekusi dengan menambahkannya pada elemen <script>.

Maka dari itu, perender tidak memiliki akses langsung ke *require* atau Node.js APIs yang lain. Untuk menyertakan modul NPM secara langsung di perender, kita harus menggunakan *bundler toolchains* yang sama (misalnya, webpack atau parcel) yang digunakan di web.

Untuk cara membuat aplikasi dengan *electron* adalah sebagai berikut : [4]

1. **Prasyarat**
   - Install terlebih dahulu Node.js dan npm (pakai versi terakhir LTS yang tersedia).
   - Cek versinya dengan mengetik node -v dan npm -v pada command prompt.
2. **Kerangka Projek**
   - Buat folder dan inisialisasi paket npm dengan mengetik mkdir namaFolder lalu ketik cd namaFolder setelah itu ketik npm init pada *command prompt*. Maka isi *package.json* akan seperti pada Kode 2.3.

Kode 2.3: Package.json

```
{
    "name": "my-electron-app",
    "version": "1.0.0",
    "description": "Hello World!",
    "main": "main.js",
    "author": "Jane Doe",
    "license": "MIT",
}
```

   - Install aplikasi *electron* ke dalam folder yang telah dibuat dengan cara mengetik npm install –save-dev electron pada command prompt.
   - Pada *script file* package.json tambahkan *command start* seperti pada Kode 2.4.

Kode 2.4: *Start command*

```
{
    "scripts": {
        "start": "electron ."
    }
}
```

   - Jalankan aplikasi *electron* dengan mengetik *npm start* pada *command prompt*. (Namun, pada tahap ini akan muncul *error* yang memberi tahu bahwa *electron* tidak dapat menemukan aplikasi untuk dijalankan.)
3. **Jalankan proses utama**
   - Buatlah file kosong bernama *main.js* di folder *root* proyek. (Jika kita mengetikan kembali *npm start* pada *command prompt*, maka aplikasi *Electron* akan berjalan dan sudah tidak ada *error*.

4. **Buat halaman web**
   Sebelum kita dapat membuat jendela untuk aplikasi kita, kita perlu membuat konten yang akan dimuat ke dalamnya. Di *Electron*, setiap jendela menampilkan konten web yang dapat dimuat dari *file HTML* lokal.
   - Buatlah *file index.html* di folder *root* projek. Contoh kode *index.html* dapat dilihat pada Kode 2.5.

Kode 2.5: Contoh kode *index.html*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="Content-Security-Policy" content="default-src␣'self';␣script-src␣'self'">
    <meta http-equiv="X-Content-Security-Policy" content="default-src␣'self';␣script-src␣'self'">
    <title>Hello World!</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    We are using Node.js <span id="node-version"></span>,
    Chromium <span id="chrome-version"></span>,
    and Electron <span id="electron-version"></span>.
  </body>
</html>
```

5. **Membuka halaman web di jendela browser**
   Setelah kita memiliki halaman web, untuk dapat memuatnya ke dalam jendela aplikasi, kita memerlukan dua modul *Electron*. Pertama adalah modul *app*, yang berfungsi untuk mengontrol application's event lifecycle. Kedua adalah modul *BrowserWindow*, yang berfungsi untuk membuat dan mengelola jendela aplikasi.
   - Karena proses utama menjalankan *Node.js*, kita dapat mengimpor Kode 2.6 sebagai modul *CommonJS* di bagian atas *file main.js*.

Kode 2.6: Kode impor modul *CommondJS*

```
const { app, BrowserWindow } = require('electron')
```

   - Tambahkan fungsi *createWindow()* yang memuat *index.html* ke dalam *BrowserWindow* baru dengan menambahkan Kode 2.7.

Kode 2.7: Fungsi CreateWindow

```
const createWindow = () => {
  const win = new BrowserWindow({
    width: 800,
    height: 600
  })

  win.loadFile('index.html')
}
```

   - Panggil fungsi createWindow() ini untuk membuka jendela. Di *Electron*, jendela browser hanya dapat dibuat setelah *event* aplikasi modul diaktifkan. Kita bisa menunggu event ini dengan menggunakan *app.whenReady() API*. Panggil *createWindow()* setelah *whenReady()* menyelesaikan *Promise*nya dengan menambahkan Kode 2.8.

Kode 2.8: Memanggil fungsi createWindow

```
app.whenReady().then(() => {
  createWindow()
})
```

6. **Kelola siklus hidup jendela**
   - Perhatikan event modul *app 'window-all-closed'* dan panggil *app.quit()* jika pengguna tidak menggunakan *macOS (darwin)* dengan menambahkan Kode 2.9.

Kode 2.9: Memanggil fungsi quit

```
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') app.quit()
})
```

7. **Akses Node.js dari perender dengan *preload script***
   Hal terakhir yang harus dilakukan adalah mencetak nomor versi untuk *Electron* dan dependensinya ke halaman web. *Preload script* berjalan sebelum proses perender dimuat, dan memiliki akses ke perender global (jendela dan dokumen) dan lingkungan *Node.js*.
   - Buat skrip baru bernama *preload.js* yang berisi seperti Kode 2.10.

Kode 2.10: Kode *preload.js*

```
window.addEventListener('DOMContentLoaded', () => {
  const replaceText = (selector, text) => {
    const element = document.getElementById(selector)
    if (element) element.innerText = text
  }

  for (const dependency of ['chrome', 'node', 'electron']) {
    replaceText(`${dependency}-version`, process.versions[dependency])
  }
})
```

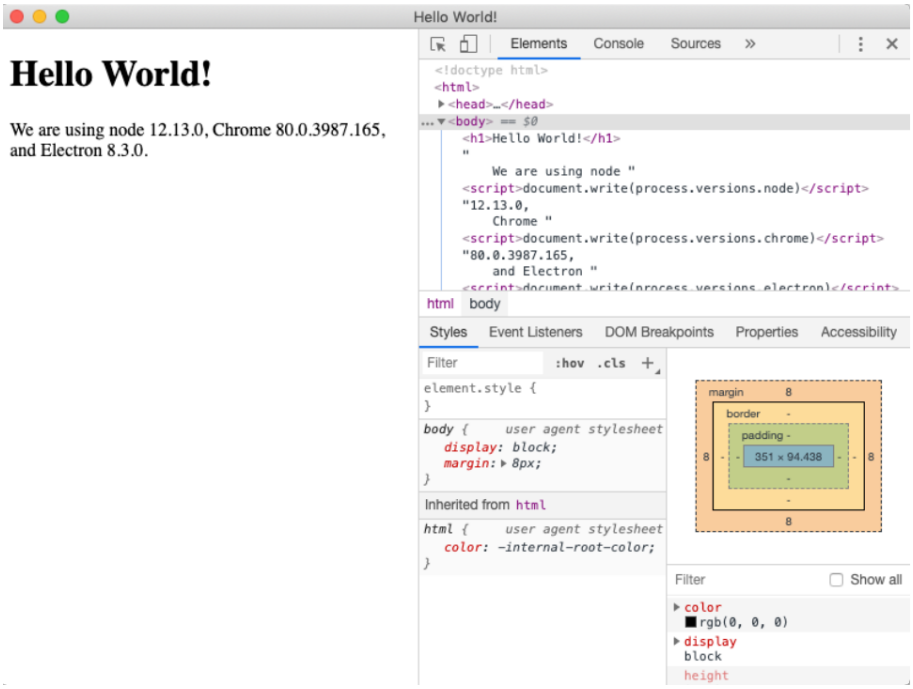
Kode 2.10 mengakses objek *process.versions Node.js* dan menjalankan fungsi dasar pembantu *replaceText* untuk memasukkan nomor versi ke dalam dokumen *HTML*.

- Untuk melampirkan skrip ini ke proses perender, sampaikan jalur ke *preload script* ke opsi *webPreferences.preload* di konstruktor *BrowserWindow* dengan menambahkan Kode 2.11 pada *file main.js*.

Kode 2.11: Penambahan kode pada *file main.js*

```
// include the Node.js 'path' module at the top of your file
const path = require('path')

// modify your existing createWindow() function
const createWindow = () => {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js')
    }
  })

  win.loadFile('index.html')
}
// ...
```

Ada dua konsep Node.js yang digunakan, pertama adalah `__dirname` yang menunjuk ke jalur skrip yang sedang dieksekusi (dalam hal ini folder root proyek). Kedua adalah *path.join API* yang menggabungkan beberapa segmen jalur bersama-sama, membuat *string* jalur gabungan yang berfungsi di semua platform.
Setelah mengikuti seluruh langkah ini, maka hasilnya akan seperti Gambar 2.17



Gambar 2.17: Contoh membuat program *Electron*

2.3 Vis.js

Vis.js adalah sebuah visualisasi *library* berbasis *browser* yang dinamis. *Library* dirancang agar mudah digunakan, untuk menangani sejumlah besar data dinamis, dan memungkinkan untuk manipulasi dan berinteraksi dengan data. *Library* tersebut terdiri dari komponen *DataSet*, *Timeline*, *Network*, *Graph2d* dan *Graph3d*. [5]

2.3.1 Timeline

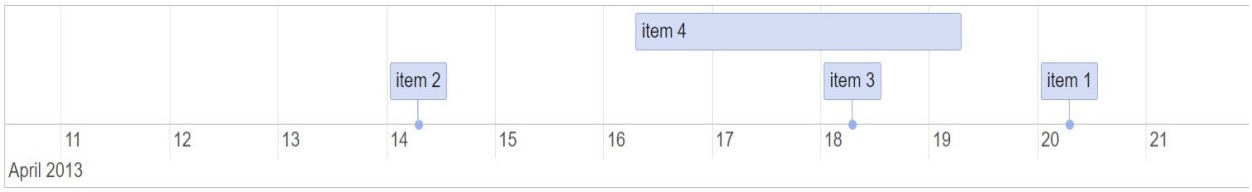
Timeline adalah grafik visualisasi interaktif untuk memvisualisasikan data dalam bentuk waktu. Item data dapat berlangsung pada satu tanggal, atau memiliki tanggal mulai dan berakhir. Kita dapat dengan bebas memindahkan dan memperbesar *timeline* dengan menyeret dan menggulir di *timeline*. Item dapat dibuat, diedit, dan dihapus di *timeline*. Skala waktu pada sumbu disesuaikan secara otomatis, dan mendukung skala mulai dari milidetik hingga tahun. *Timeline* menggunakan HTML DOM biasa untuk merender *timeline* dan item yang diletakkan di *timeline*. Hal ini memungkinkan penyesuaian yang fleksibel menggunakan *css style*. [6]

Untuk menggunakan *timeline*, kita harus menyertakan file *vis.js* dan *vis-timeline-graph2d.min.css* yang dapat diunduh dari visjs.org. Contoh kode berikut pembuatan *timeline* dapat dilihat pada Kode 2.12 dan untuk hasilnya dapat dilihat pada gambar 2.18.

Kode 2.12: Contoh kode untuk membuat *timeline* menggunakan *vis.js*

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Timeline | Basic demo</title>

  <style type="text/css">
    body, html {
      font-family: sans-serif;
    }
  </style>

  <script src="../../dist/vis.js"></script>
  <link href="../../dist/vis-timeline-graph2d.min.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="visualization"></div>

```

```
<script type="text/javascript">
  // DOM element where the Timeline will be attached
  var container = document.getElementById('visualization');

  // Create a DataSet (allows two way data-binding)
  var items = new vis.DataSet([
    {id: 1, content: 'item 1', start: '2013-04-20'},
    {id: 2, content: 'item 2', start: '2013-04-14'},
    {id: 3, content: 'item 3', start: '2013-04-18'},
    {id: 4, content: 'item 4', start: '2013-04-16', end: '2013-04-19'},
    {id: 5, content: 'item 5', start: '2013-04-25'},
    {id: 6, content: 'item 6', start: '2013-04-27'}
  ]);

  // Configuration for the Timeline
  var options = {};

  // Create a Timeline
  var timeline = new vis.Timeline(container, items, options);
</script>
</body>
</html>
```



Gambar 2.18: Hasil contoh untuk membuat *timeline* menggunakan *vis.js*

2.3.2 Network

Jaringan (*Network*) adalah visualisasi untuk menampilkan jaringan - jaringan yang terdiri dari *node* dan *edge*. Visualisasinya mudah digunakan dan mendukung bentuk kustom, gaya, warna, ukuran, gambar, dan lain - lain. Visualisasi jaringan bekerja dengan lancar di *browser* moderen apapun hingga beberapa ribu *node* dan *edge*. Untuk menangani jumlah node yang lebih besar, jaringan memiliki dukungan pengelompokan. Jaringan menggunakan *HTML canvas* untuk *rendering*. [7]

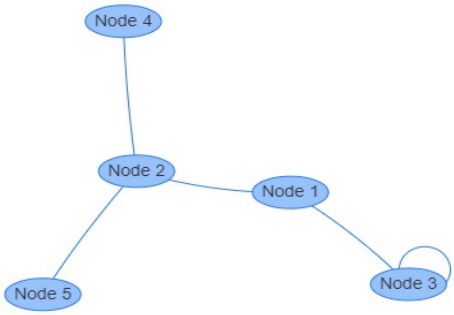
Untuk menggunakan *vis-network*, kita harus menyertakan file *vis.js* dan *vis-network.min.css* yang dapat diunduh dari visjs.org, atau dengan tautkan dari unpkg.com. Jika kita menambahkan ini ke aplikasi kita, kita perlu menentukan *node* dan *edgenya*. Kita juga dapat menggunakan *vis.DataSets* untuk pengikatan data dinamis, misalnya, mengubah warna, label, atau pilihan apapun setelah kita menginisialisasi jaringan.

Setelah kita memiliki data, yang kita butuhkan hanyalah *container div* untuk memberi tahu *vis* di mana harus meletakkan jaringan kita. Selain itu, kita dapat menggunakan pilihan objek untuk menyesuaikan banyak aspek jaringan. Contoh kode pembuatan *network* dimana file *vis-network.js* dimasukkan dengan cara menautkan dari unpkg.com dapat dilihat pada Kode 2.13 dan untuk hasilnya dapat dilihat pada gambar 2.19.

Kode 2.13: Contoh kode untuk membuat *network* menggunakan *vis.js*

```
<!DOCTYPE HTML>
<html>
<head>
    <script type="text/javascript" src="https://unpkg.com/vis-network/standalone/umd/vis-network.min.js"></script>

    <style type="text/css">
        #mynetwork {
            width: 600px;
            height: 400px;
            border: 1px solid lightgray;
        }
    </style>
</head>
<body>
<div id="mynetwork"></div>

<script type="text/javascript">
    // create an array with nodes
    var nodes = new vis.DataSet([
        {id: 1, label: 'Node 1'},
        {id: 2, label: 'Node 2'},
        {id: 3, label: 'Node 3'},
```

```
        {id: 4, label: 'Node 4'},
        {id: 5, label: 'Node 5'}
    ]);

    // create an array with edges
    var edges = new vis.DataSet([
        {from: 1, to: 3},
        {from: 1, to: 2},
        {from: 2, to: 4},
        {from: 2, to: 5}
    ]);

    // create a network
    var container = document.getElementById('mynetwork');

    // provide the data in the vis format
    var data = {
        nodes: nodes,
        edges: edges
    };
    var options = {};

    // initialize your network!
    var network = new vis.Network(container, data, options);
</script>
</body>
</html>
```



Gambar 2.19: Hasil contoh untuk membuat *network* menggunakan *vis.js*

2.3.3 DataSet

Vis.js hadir dengan *DataSet* yang fleksibel, yang dapat digunakan untuk menyimpan dan memanipulasi data yang tidak terstruktur serta memperhatikan perubahan dalam data tersebut. *DataSet* berbasis nilai, dimana item data dapat ditambahkan, diperbarui, dihapus, dan seseorang dapat menyetujui perubahan tersebut dalam *DataSet*. *DataSet* juga dapat digunakan untuk menyimpan objek *JSON* berdasarkan idnya. Objek dapat ditambahkan, diperbarui, dan dihapus dari *DatSet*. Data dalam *DataSet* dapat difilter dan diurutkan, serta *fields* (seperti tanggal) dapat dikonversi ke dalam tipe tertentu. Data dapat dinormalisasi saat menambahkannya kedalam *DataSet*. [8]

Terdapat beberapa *method* yang telah disediakan, seperti :

- add ()
  Method ini berfungsi untuk menambahkan *item*, dimana item data dapat berisi properti dan format data yang berbeda.
- updateOnly ()
  Method ini berfungsi untuk memperbarui *item* yang sudah ada.
- remove ()
  Method ini bergfungsi untuk menghapus *item*.
- get ()
  Method ini berfungsi untuk mendapatkan *item* tertentu.

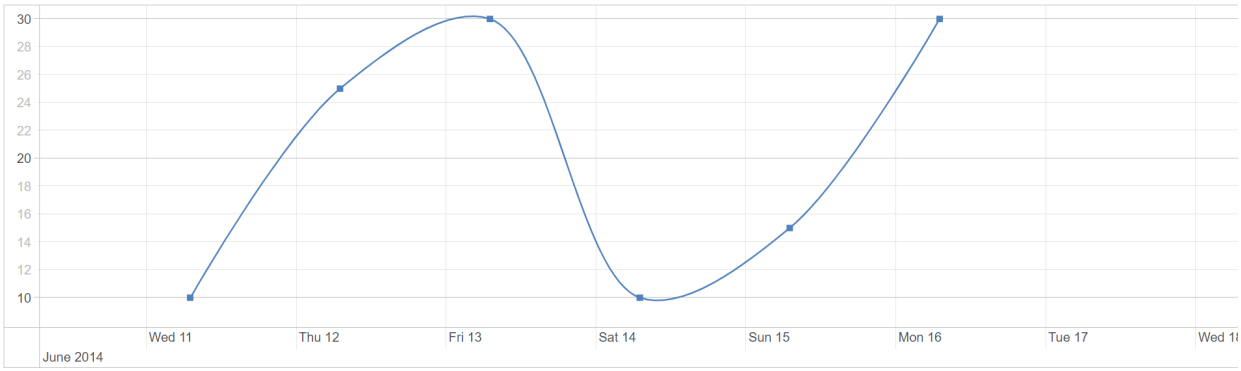
2.3.4 Graph2d

Graph2d adalah grafik visualisasi interaktif yang berfungsi untuk menggambar data dalam bentuk grafik dua dimensi. Kita dengan bebas dapat memindahkan dan memperbesar grafik dengan cara menyeret dan menggulir pada jendela grafik. *Graph2d* menggunakan *HTML DOM* (*JavaScript*) dan *SVG* (*Scalable Vector Graphics*) untuk proses *rendering*. Hal ini memungkinkan penyesuaian yang fleksibel menggunakan *styling* css. [9]

Contoh kode pembuatan *Graph2d* dapat dilihat pada Kode 2.14 dan untuk hasilnya dapat dilihat pada gambar 2.20.

Kode 2.14: Contoh kode untuk membuat *graph2d* menggunakan *vis.js*

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Graph2d or Basic Example</title>

  <style type="text/css">
    body, html {
      font-family: sans-serif;
    }
  </style>

  <script src="../../dist/vis.js"></script>
  <link href="../../dist/vis-timeline-graph2d.min.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="visualization"></div>

<script type="text/javascript">
  var container = document.getElementById('visualization');
  var items = [
      {x: '2014-06-11', y: 10},
      {x: '2014-06-12', y: 25},
      {x: '2014-06-13', y: 30},
      {x: '2014-06-14', y: 10},
      {x: '2014-06-15', y: 15},
      {x: '2014-06-16', y: 30}
  ];

  var dataset = new vis.DataSet(items);
  var options = {
      start: '2014-06-10',
      end: '2014-06-18'
  };
  var Graph2d = new vis.Graph2d(container, dataset, options);
</script>
</body>
</html>
```



Gambar 2.20: Hasil contoh untuk membuat *graph2d* menggunakan *vis.js*

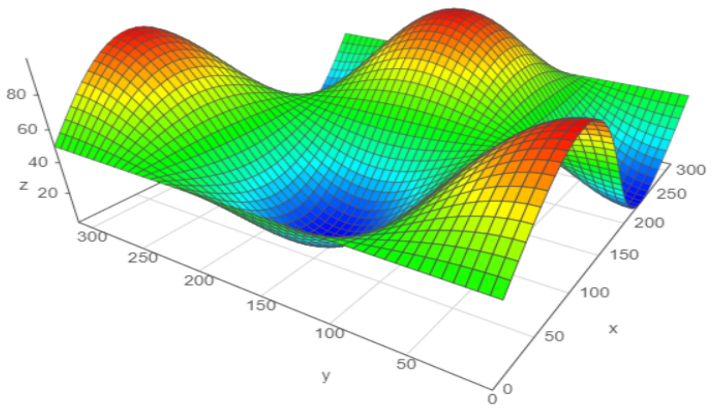
2.3.5 Graph3d

Graph3d adalah grafik visualisasi interaktif yang berfungsi untuk menggambar data dalam bentuk grafik tiga dimensi. Kita dengan bebas dapat memindahkan dan memperbesar grafik dengan cara menyeret dan menggulir pada jendela grafik. Graph3d juga mendukung animasi grafik serta menggunakan HTML canvas untuk membuat grafik, dan dapat merender hingga beberapa ribu titik data dengan lancar. [10]

Contoh kode pembuatan *Graph2d* dapat dilihat pada Kode 2.15 dan untuk hasilnya dapat dilihat pada gambar 2.21.

Kode 2.15: Contoh kode untuk membuat *graph3d* menggunakan *vis.js*

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Graph 3D demo</title>

  <style>
    body {font: 10pt arial;}
  </style>

  <script type="text/javascript" src="../../dist/vis.js"></script>

  <script type="text/javascript">
  var data = null;
  var graph = null;

  function custom(x, y) {
    return (Math.sin(x/50) * Math.cos(y/50) * 50 + 50);
  }

  // Called when the Visualization API is loaded.
  function drawVisualization() {
    // Create and populate a data table.
    var data = new vis.DataSet();
    // create some nice looking data with sin/cos
    var steps = 50;  // number of datapoints will be steps*steps
    var axisMax = 314;
    var axisStep = axisMax / steps;
    for (var x = 0; x < axisMax; x+=axisStep) {
      for (var y = 0; y < axisMax; y+=axisStep) {
        var value = custom(x, y);
        data.add({
          x: x,
          y: y,
          z: value,
          style: value
        });
      }
    }

    // specify options
    var options = {
      width:  '600px',
      height: '600px',
      style: 'surface',
      showPerspective: true,
      showGrid: true,
      showShadow: false,
      keepAspectRatio: true,
      verticalRatio: 0.5
    };

    // create a graph3d
    var container = document.getElementById('mygraph');
    graph3d = new vis.Graph3d(container, data, options);
  }
  </script>
</head>

<body onload="drawVisualization();">
  <div id="mygraph"></div>
</body>
</html>
```



Gambar 2.21: Hasil contoh untuk membuat *graph3d* menggunakan *vis.js*

2.4 FTIS Open Data

FTIS (Fakultas Teknologi Informasi dan Sains) adalah sebuah fakultas milik UNPAR (Universitas Katolik Parahyangan) yang memiliki tiga program studi, yaitu : Matematika, Fisika, dan Teknik Informatika. *Open data* adalah data yang dapat bebas digunakan oleh semua orang untuk digunakan dan diterbitkan kembali sesuai keinginan, tanpa adanya batasan hak cipta, paten, atau mekanisme kontrol lainnya. Maka dari itu FTIS *Open Data* dapat diartikan sebagai data milik FTIS khususnya program studi teknik informatika yang secara bebas dapat digunakan secara bebas oleh orang lain. FTIS *open data* tersebut dapat diakses pada https://ftisunpar.github.io/data/prasyarat.json. [11]

Endpoint didefinisikan sebagai *url* yang mengikuti setelah basis *url* https://ftisunpar.github.io/data/. Sebagai contoh *endpoint* prasyarat.json memiliki alamat *url* lengkap https://ftisunpar.github.io/data/prasyarat.json. Prasyarat.json memiliki informasi tentang seluruh matakuliah, jumlah SKS, posisi semester, serta prasyarat yang ada. Dengan bentuk datanya adalah array untuk setiap matakuliahnya. Setiap struktur matakuliah memiliki properti sebagai berikut:

- kode (String)
- nama (String)
- prasyarat
  - tempuh (String[])
  - lulus (String[])
  - bersamaan (String[])
  - berlakuAngkatan (Number|null)
- sks (Number)
- wajib (Boolean)
- semester (Number)

Untuk lebih jelasnya dapat dilihat pada Kode 2.16.

Kode 2.16: prasyarat.json

```
{
    "kode": "AIF181100",
    "nama": "Dasar Pemrograman",
    "prasyarat": {
      "tempuh": [],
      "lulus": [
        "AIF181101"
      ],
      "bersamaan": [],
      "berlakuAngkatan" : 2018
    },
    "sks": 4,
    "wajib": true,
    "semester": 2
}
```

Pada prasyarat lulus, diberikan sebuah string berupa kode mata kuliah yang menjadi prasyarat lulus untuk mata kuliah tersebut. Definisi prasyarat berdasarkan macamnya :

- Tempuh
  Mahasiswa diharuskan sudah pernah menempuh mata kuliah yang disebutkan.
- Lulus
  Mahasiswa diharuskan untuk lulus mata kuliah yang disebutkan.
- Bersamaan
  Mata kuliah tersebut harus di ambil secara bersamaan dengan mata kuliah yang disebutkan (butuh informasi lagi).
- Berlaku Angkatan
  Property ini mulai berlaku karena pergantian kurikulum. Prasyarat ini memiliki maksud bahwa matakuliah ini mulai berlaku semenjak angkatan x.

2.5 JavaScript

JavaScript (sering disingkat menjadi JS) adalah sebuah bahasa berorientasi objek yang dikenal sebagai bahasa *scripting* untuk halaman web, tetapi digunakan di banyak lingkungan *non-browser* juga. *Javascript* merupakan bahasa *scripting multi-paradigma* berbasis prototipe yang dinamis, dan mendukung gaya pemrograman berorientasi objek, imperatif, dan fungsional. [12]

JavaScript berjalan di sisi klien *web*, yang dapat digunakan untuk merancang / memprogram bagaimana perilaku halaman web saat terjadi suatu *event* tertentu. *JavaScript* adalah bahasa yang mudah dipelajari dan juga merupakan bahasa *scripting* yang *powerful*, karena banyak digunakan untuk mengontrol perilaku halaman web.

Berlawanan dengan kesalahpahaman pada umumnya, *JavaScript* bukanlah "*Java* yang diinterpretasikan". Singkatnya, *JavaScript* adalah bahasa *scripting* dinamis yang mendukung konstruksi objek berbasis prototipe. Sintaks dasarnya sengaja mirip dengan *Java* dan *C++* untuk mengurangi jumlah konsep baru yang diperlukan untuk mempelajari bahasa tersebut. Konstruksi bahasa, seperti pernyataan *if*, *loop for and while*, dan *switch and try ... catch blocks* berfungsi sama seperti pada bahasa ini (atau hampir sama).

JavaScript dapat berfungsi sebagai bahasa prosedural yang berorientasi objek. Objek dibuat secara terprogram dalam *JavaScript*, dengan melampirkan metode dan properti ke objek yang kosong pada waktu proses, yang bertentangan dengan sintaks kelas pada umumnya dalam bahasa yang *dicompile* seperti *C++* dan *Java*. Setelah objek telah dibangun dapat digunakan sebagai prototipe untuk membuat objek serupa.

2.5.1 Async and Await

Penambahan yang lebih baru pada bahasa *JavaScript* adalah fungsi *async* dan kata kunci *await*, ditambahkan dalam *ECMAScript* 2017. Fitur-fitur ini membuat kode *asinkron* lebih mudah untuk ditulis dan dibaca setelahnya. [13]

Fungsi *async* adalah fungsi yang mengetahui bagaimana mengharapkan kemungkinan kata kunci *await* digunakan untuk memanggil *asynchronous code*. Nilai pengembaliannya dijamin akan dikonversi menjadi *promises*. Untuk dapat menggunakan nilai yang dikembalikan saat *promise* terpenuhi, kita bisa menggunakan .then() block:. Jadi kata kunci async ditambahkan ke fungsi untuk memberi tahu mereka agar mengembalikan *promises* daripada langsung mengembalikan nilainya.

Keuntungan dari fungsi *async* hanya menjadi jelas ketika menggabungkannya dengan kata kunci *await*. *Await* hanya bekerja di dalam fungsi *async* dalam kode *JavaScript* biasa, namun dapat digunakan sendiri dengan modul *JavaScript*. *Await* dapat diletakkan di depan fungsi berbasis *promise async* apapun untuk menjeda kode pada baris itu hingga *promise* terpenuhi, lalu mengembalikan nilai yang dihasilkan. *Await* dapat digunakan saat memanggil fungsi apapun yang mengembalikan *promise*, termasuk fungsi *API web*.

Untuk membuat permintaan dan mengambil sumber daya, gunakan metode *fetch()*. Metode ini diimplementasikan di beberapa antarmuka, khususnya *Window* dan *WorkerGlobalScope*. Metode fetch() mengambil satu argumen wajib, jalur ke sumber daya yang ingin diambil. Kemudian mengembalikan promise yang memutuskan ke respon untuk permintaan itu (segera setelah server merespons dengan header) bahkan jika respons server di *HTTP* adalah status eror.

BAB 3

ANALISIS

Pada bab ini akan dijelaskan mengenai analisis bentuk data, analisis bentuk visualisasi, analisis sistem visualisasi, dan perancangan modul.

3.1 Analisis Bentuk Data

Untuk dapat memvisualiasikan kurikulum 2018, diperlukan sumber data kurikulum 2018 yang terdapat pada github API. Untuk pengambilan datanya terdapat pada variable const fetchUsers dimana terdapat fungsi *asynchronous* dengan kata kunci await untuk memanggil fungsi API web yang adalah fungsi *fetch*.

3.2 Analisis Bentuk Visualisasi

Vis.js memiliki 4 buah jenis visualisasi yaitu *network*, *timeline*, *graph2d* dan *graph3d*. Pada subbab ini akan dijelaskan jenis visualisasi apa yang dapat digunakan untuk memvisualisasikan kurikulum 2018.

3.2.1 Graph2d

Graph2d menampilkan grafik dua dimensi, dimana terdapat dimensi horizontal (X) dan dimensi vertikal (Y). Sehingga gambar hanya dapat digeser ke kanan atau kiri dan atas atau bawah. Maka dari itu, *grapgh2d* cocok untuk memvisualisasikan data yang bersifat *continue* sedangkan, untuk data kurikulum 2018 kurang cocok karena bersifat diskrit.

3.2.2 Graph3d

Graph3d menampilkan grafik tiga dimensi, dimana terdapat dimensi horizontal (X), dimensi vertikal (Y), dan dimensi kedalaman (Z). Sehingga gambar dapat melakukan rotasi dari berbagai perspektif. Maka dari itu, *graph3d* kurang cocok untuk memvisualisasikan data kurikulum 2018 karena datanya tidak dapat direpresentasikan dalam bentuk ketiga dimensi tersebut.

3.2.3 Timeline

Timeline menampilkan data dalam waktu, dimana data dapat berlangsung pada satu tanggal, atau memiliki tanggal mulai dan berakhir (rentang). Maka dari itu, *timeline* cocok untuk membuat pohon kurikulum 2018, dimana kode matakuliah dapat dimodelkan dalam bentuk *id*. Kemudian untuk nama matakuliah dapat dimodelkan dalam bentuk *content* yang berbentuk kotak dengan warna :

- biru muda untuk setiap matakuliah yang terdapat pada semester satu,
- kuning untuk setiap matakuliah yang terdapat pada semester dua,
- merah untuk setiap matakuliah yang terdapat pada semester tiga,
- hijau untuk setiap matakuliah yang terdapat pada semester empat,

- magenta untuk setiap matakuliah yang terdapat pada semester lima,
- ungu untuk setiap matakuliah yang terdapat pada semester enam,
- oranye untuk setiap matakuliah yang terdapat pada semester tujuh,
- biru tua untuk setiap matakuliah yang terdapat pada semester delapan.

Setiap *content* akan dibagi kedalam dua grup menjadi matakuliah wajib yang terletak pada bagian atas *timeline* dan matakuliah pilihan yang terletak pada bagian bawah *timeline*. Visualisasi ini hanya akan memvisualisasikan lamanya masa kuliah yang seharusnya, yaitu selama empat tahun yang terdiri dari delapan semester. Untuk semester ganjil akan dimulai dari bulan Agustus sampai bulan Desember dan untuk semester genap akan dimulai dari bulan Januari sampai bulan Juli. Maka dari itu, besar *content* akan mengikuti waktu untuk setiap semesternya.

3.2.4 Network

Network menampilkan banyak jaringan yang terdiri dari banyak *node* dan banyak *edge*. Maka dari itu, *network* cocok untuk membuat pohon kurikulum 2018, dimana kode matakuliah dapat dimodelkan dalam bentuk *id*. Kemudian nama matakuliah dapat dimodelkan dengan label yang terdapat pada *node* dengan warna :

- biru muda untuk setiap matakuliah yang terdapat pada semester satu,
- kuning untuk setiap matakuliah yang terdapat pada semester dua,
- merah untuk setiap matakuliah yang terdapat pada semester tiga,
- hijau untuk setiap matakuliah yang terdapat pada semester empat,
- magenta untuk setiap matakuliah yang terdapat pada semester lima,
- ungu untuk setiap matakuliah yang terdapat pada semester enam,
- oranye untuk setiap matakuliah yang terdapat pada semester tujuh,
- biru tua untuk setiap matakuliah yang terdapat pada semester delapan.

Node juga akan berbentuk kotak untuk mata kuliah wajib dan berbentuk lingkaran untuk mata kuliah pilihan. Untuk matakuliah a yang memiliki prasyarat b dapat dimodelkan dengan *edges* berarah dari *node* b ke *node* a. Terdapat tiga macam edges yang warnanya mengikuti warna *node*nya, yaitu :

1. edges yang berbentuk garis dengan ujung anak panah yang melambangkan prasyarat tempuh.
2. edges yang berbentuk garis putus - putus dengan ujung anak panah yang melambangkan prasyarat lulus.
3. edges yang berbentuk garis putus - putus saja yang melambangkan prasyarat bersamaan.

Terdapat delapan buah node tambahan yang berisikan semester satu sampai semester delapan secara naik berurutan ke bawah yang berfungsi untuk mengelompokkan *node* - *node* untuk setiap semesternya.

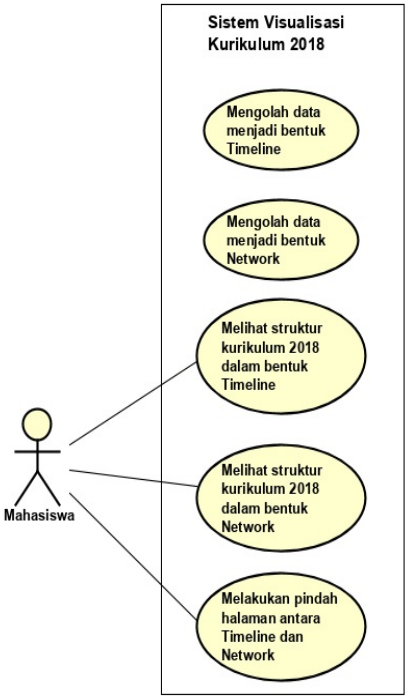
3.3 Analisis Sistem Visualisasi

3.3.1 Spesifikasi Kebutuhan Perangkat Lunak

Sesuai dengan rumusan masalah, perangkat lunak yang dibangun hanya akan mengolah data menjadi bentuk visualisasi yang paling cocok untuk memvisualisasikan kurikulum 2018. Perangkat lunak yang dibangun akan menggunakan *framework Electron* dan pustaka *Vis.js*. Pertama - tama perangkat lunak akan mengambil data kurikulum 2018 dari *API*. Kemudian data yang diambil akan diolah kedalam bentuk visualisasi *Network* dan *Timeline*, dimana kedua bentuk visualisasi tersebut merupakan bentuk yang paling cocok untuk memvisualisasikan kurikulum 2018.

Perangkat lunak ini dapat berjalan di berbagai sistem operasi. Sesuai dengan kemampuan framework *Electron* yang merupakan aplikasi lintas platform, maka pada penelitian ini akan menggunakan sistem operasi *Windows*. Namun, perangkat lunak akan tetap dapat berjalan pada sistem operasi *Linux* maupun *MacOs*.

3.3.2 Use Case Diagram



Gambar 3.1: Use Case Diagram

Pada diagram *use case* (gambar : 3.1) terdapat tiga fitur bagi mahasiswa yaitu, melihat struktur kurikulum 2018 dalam bentuk *Timeline*, melihat struktur kurikulum 2018 dalam bentuk *Network*, dan melakukan pindah halaman antara *Timeline* dan *Network*. Selanjutnya bagi sistem, terdapat 2 fitur yaitu, mengolah data dalam bentuk *Timeline* dan mengolah data menjadi bentuk *Network*.

Scenario Use Case

- Sistem akan mengambil data dari https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json.
- Sistem mengolah data menjadi bentuk *Timeline*.
- Sistem mengolah data menjadi bentuk *Network*.
- Mahasiswa melihat struktur kurikulum 2018 dalam bentuk *Timeline*.
- Mahasiswa melihat struktur kurikulum 2018 dalam bentuk *Network*.
- Mahasiswa dapat melakukan pindah halaman antara halaman *Timeline* dengan halaman *Network*.

3.3.3 Perancangan Modul

- Pada berkas *JavaScript network* terdiri dari:
  - **fungsi *processNetwork***, dimana merupakan fungsi utama yang akan menjalankan fungsi - fungsi lainnya, seperti :
    - **fungsi *fetchUsers***, yang akan menjalankan fungsi *async* dengan kata kunci await ditambah dengan fungsi *fetch* yang berfungsi untuk meminta dan mengambil data dari API. Untuk kodenya dapat dilihat pada Kode 3.1.

Kode 3.1: Kode pengambilan data dari API

```
let matkul = '';
const fetchUsers = async () => {
    try {
        const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat
            .json');
        if (!res.ok) {
            throw new Error(res.status);
        }
        const data = await res.json();
        matkul = data;
        createNetwork();
    } catch (error) {
        console.log(error);
    }
}
```

* **fungsi *createNetwork***, yang berfungsi untuk mengolah data yang telah diambil dari API menjadi sebuah visualisasi yang berbentuk *network*. Di dalam fungsi ini terdapat sebuah *looping* untuk membuat delapan buah *node* yang berisikan semester satu sampai dengan semester delapan yang berfungsi sebagai patokan untuk penempatan setiap node mata kuliah sesuai dengan letak semesternya. Selanjutnya terdapat sebuah *looping* untuk membuat *node* - *node* sesuai dengan jumlah mata kuliah, dimana untuk bentuk dan posisi *node*nya akan diatur di dalamnya. Pada *looping* ini juga akan dibuatkan *edges* untuk setiap *node*nya sesuai dengan prasyaratnya masing - masing, dimana untuk bentuk *edges*nya akan diatur di dalamnya. Tidak ada fungsi khusus yang dibuat untuk memberi warna pada setiap *node* yang ada, karena warna *node* akan diberikan secara otomatis oleh *library Vis.js*. Untuk kodenya dapat dilihat pada Kode 3.2.

Kode 3.2: Kode untuk membuat Visualisasi Network

```
const horizontal = [];
for (i = 1; i < 9; i++) {
    nodes.add([{
        id: i, label: "Semester " + i, group: i, y: i*150
    }])
    if (i != 8) {
        edges.add([
            { from: i, to: i + 1 }
        ])
    }
    horizontal[i] = 0;
}

for (let i = 0; i < matkul.length; i++) {
    let shape=""
    if(matkul[i].wajib){
        shape="box"
    }
    else{
        shape="circle"
    }

    horizontal[matkul[i].semester] += 200;
    nodes.add([{
        id: matkul[i].kode, label: matkul[i].nama, group: matkul[i].semester,
        x:horizontal[matkul[i].semester], y: matkul[i].semester*150, shape:shape
    }])

    if(matkul[i].prasyarat.tempuh.length != 0){
        for (let j = 0; j < matkul[i].prasyarat.tempuh.length; j++){
            edges.add([
                { from: matkul[i].kode, to: matkul[i].prasyarat.tempuh[j],
                  arrows: "from"}
            ])
        }
    }
    if(matkul[i].prasyarat.lulus.length != 0){
        for (let j = 0; j < matkul[i].prasyarat.lulus.length; j++){
            edges.add([
                { from: matkul[i].kode, to: matkul[i].prasyarat.lulus[j],
                  dashes: true }
            ])
        }
    }
    if(matkul[i].prasyarat.bersamaan.length != 0){
        for (let j = 0; j < matkul[i].prasyarat.bersamaan.length; j++){
            edges.add([
                { from: matkul[i].kode, to: matkul[i].prasyarat.bersamaan[j],
                  arrows: "from", dashes: true }
            ])
        }
    }
}
```

- Pada berkas *javascript timeline* terdiri dari:
  - **fungsi *processTimeline***, dimana merupakan fungsi utama yang akan menjalankan fungsi - fungsi lainnya, seperti :
    - **fungsi *fetchUsers***, yang akan menjalankan fungsi *async* dengan kata kunci await ditambah dengan fungsi *fetch* yang berfungsi untuk meminta dan mengambil data dari API. Untuk kodenya dapat dilihat pada Kode 3.1.
    - **fungsi *createTimeline*** yang berfungsi untuk mengolah data yang telah diambil dari API menjadi sebuah visualisasi yang berbentuk *timeline*. Di dalam fungsi ini terdapat sebuah *variable groups* yang berfungsi untuk membuat dua *group* yaitu mata kuliah wajib dan mata kuliah pilihan yang akan muncul pada bagian sisi kiri (sumbu y) *timeline*. Selanjutnya terdapat fungsi *Date* untuk mengambil data tahun dan bulan saat ini, dimana akan digunakan untuk menentukan waktu yang akan muncul pada bagian sisi bawah (sumbu x) *timeline*. Selanjutnya terdapat sebuah *looping* untuk mengisi *array* semester di mana setiap arraynya akan berisi informasi tentang bulan dan tahun untuk setiap semesternya. Kemudian terdapat sebuah *looping* untuk membuat *content* sesuai dengan jumlah mata kuliah, dimana warna, ukuran, dan letak *content* akan diatur di dalamnya. Untuk kodenya dapat dilihat pada Kode 3.3.

Kode 3.3: Kode untuk membuat Visualisasi Network

```
            var groups = new vis.DataSet([
              { id: 1, content: "Mata␣Kuliah␣Wajib" },
              { id: 2, content: "Mata␣Kuliah␣Pilihan" },
            ]);

            let tahun=new Date().getFullYear();
            let bulan = new Date().getMonth();
            if(bulan<7){
              tahun-=1
              bulan=7
            }
            else{
              bulan=7
            }

            let semester=[]
            for(i=1;i<=9;i++){
                semester[i]= tahun.toString()+"-"+bulan.toString()+"-␣31"
                if(i%2!=0){
                  bulan+=5
                }
                else{
                  bulan-=5
                  tahun+=1
                }
            }

            var res = [];
            let color=["","lightBlue","yellow","red","green","pink","purple","orange","darkBlue"]
            for (var i = 0; i < matkul.length; i++) {
                let wajib=''
                if(matkul[i].wajib){
                  wajib=1;
                }
                else{
                  wajib=2;
                }
                res.push(
                  { id: matkul[i].kode, content: matkul[i].nama, editable: false,group:wajib ,
                    start: semester[matkul[i].semester],end:semester[matkul[i].semester+1],
                    className: color[matkul[i].semester]  }
                )
            }
            var items = new vis.DataSet(res);

            var container = document.getElementById('timeline');

            var options = {};

            var timeline = new vis.Timeline(container, items,groups, options);
          }
          fetchUsers();
```

BAB 4

PERANCANGAN

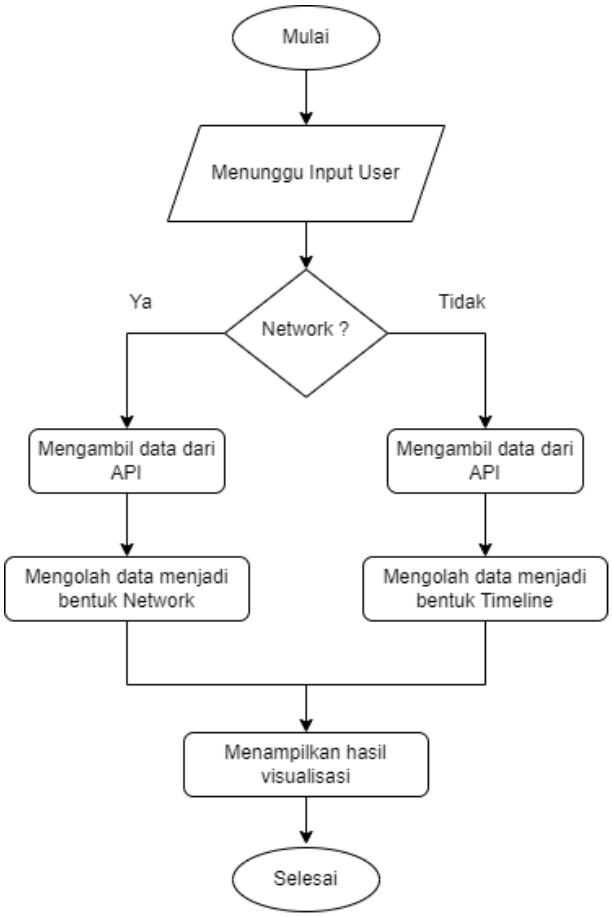
Pada bab ini akan dijelaskan mengenai perancangan perangkat lunak yang meliputi: perancangan proses visualisasi dan perancangan antarmuka

4.1 Perancangan Proses Visualisasi

Untuk Perancangan proses visualisai akan dijelaskan menggunakan flowchart diagram yang dilanjutkan dengan perancangan struktur aplikasi.

4.1.1 Flowchart Diagram

Untuk diagaram alur aplikasi VisKur dapat dilihat pada Gambar 4.1.



Gambar 4.1: *Flowchart* aplikasi VisKur

Saat pertama kali perangkat lunak dijalankan, perangkat lunak akan menunggu *input user*, apakah pengguna akan meminta dibuatkan visualisasi kurikulum 2018 dalam bentuk *Network* atau bentuk *Timeline*. Jika pengguna memlih bentuk *Network*, maka perangkat lunak akan segera mengambil data dari API, mengolah data tersebut menjadi bentuk *Network*, dan ketika sudah selesai mengolahnya, maka perangkat lunak akan menampilkan hasil visualisasi dalam bentuk *Network* tersebut. Jika pengguna tidak memilih bentuk *Network* yang artinya memilih bentuk *Timeline*, maka maka perangkat lunak akan segera mengambil data dari API, mengolah data tersebut menjadi bentuk *Timeline*, dan ketika sudah selesai mengolahnya, makan perangkat lunak akan menampilkan hasil visualisasi dalam bentuk *Timeline* tersebut. Ketika perangkat lunak sudah berhasil menampilkan hasil visualisai ke dalam bentuk yang diinginkan pengguna, maka perangkat lunak akan selesai.

4.1.2 Perancangan Struktur Aplikasi

Struktur aplikasi merupakan susunan direktori untuk membangun aplikasi tersebut. Struktur ini terdiri dari berbagai folder dan berkas yang telah dipisahkan berdasarkan fungsinya masing-masing. Berikut ini merupakan penjelasan masing-masing folder dan berkas yang digunakan untuk membuat aplikasi VisKur:

- **folder css**, folder ini berisi berkas - berkas dengan ekstensi *css* yang digunakan untuk mengatur tampilan aplikasi. Terdapat berbagai berkas pada folder ini, yaitu:
  - **vis-network.min.css**, berkas *css* ini berfungsi untuk mengatur tampilan setiap elemen *network* pada *Vis.js*.
  - **vis-timeline-graph2d.min.css**, berkas *css* ini berfungsi untuk mengatur tampilan setiap elemen *timeline* pada *Vis.js*. Pada berkas ini akan ditambahkan Kode 4.1 yang berfungsi untuk memberikan warna pada setiap *content* yang ditampilkan.

Kode 4.1: Tambahan kode *vis-timeline-graph2d.min.css*

```
.vis-item.lightBlue {
  background-color: rgb(151, 194, 252);
  border-color: rgb(43, 124, 233);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.yellow {
    background-color: rgb(255,255,0);
    border-color:rgb(255,165,0);
    color: black;
    box-shadow: 0 0 10px gray;
  }
.vis-item.red {
  background-color: rgb(251,126,129);
  border-color: rgb(251,70,74);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.green {
  background-color: rgb(123,225,65);
  border-color: rgb(76,177,19);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.pink {
  background-color: rgb(235,125,244);
  border-color: rgb(225,41,240);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.purple {
  background-color: rgb(173,133,228);
  border-color: rgb(124,41,240);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.orange {
  background-color: rgb(255,168,7);
  border-color: rgb(217,142,3);
  color: black;
  box-shadow: 0 0 10px gray;
}
.vis-item.darkBlue {
  background-color: rgb(110,110,253);
  border-color: rgb(68,35,251);
  color: black;
  box-shadow: 0 0 10px gray;
}
```

  - **vis.css**, berkas *css* ini berfungsi untuk mengatur tampilan dan animasi untuk semua jenis visualisasi yang terdapat pada *Vis.js*.
- **folder js**, folder ini berisi berkas - berkas dengan ekstensi js. Terdapat berbagai berkas pada folder ini, yaitu:
  - **network.js**, berkas *JavaScript* ini berisi fungsi khusus untuk membuat visualisai dalam bentuk *network*. Fungsi pada berkas ini nantinya akan dipanggil pada index.html. Untuk kodenya dapat dilihat pada Kode 4.2.

Kode 4.2: Kode *network.js*

```
function processNetwork (){
    document.getElementById("timeline").style.display="none"
    document.getElementById("network").style.display="inline"
    let matkul = '';
    const fetchUsers = async () => {
        try {
            const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json'
                );
            if (!res.ok) {
                throw new Error(res.status);
            }
            const data = await res.json();
            matkul = data;
            createNetwork();
        } catch (error) {
            console.log(error);
        }
    }

    var nodes = new vis.DataSet();

    var edges = new vis.DataSet();

    function createNetwork() {
        const horizontal = [];
        for (i = 1; i < 9; i++) {
            nodes.add([{
                id: i, label: "Semester " + i, group: i, y: i*150
            }])
            if (i != 8) {
                edges.add([
                    { from: i, to: i + 1 }
                ])
            }
            horizontal[i] = 0;
        }

        for (let i = 0; i < matkul.length; i++) {
            let shape=""
            if(matkul[i].wajib){
                shape="box"
            }
            else{
                shape="circle"
            }

            horizontal[matkul[i].semester] += 200;
            nodes.add([{
                id: matkul[i].kode, label: matkul[i].nama, group: matkul[i].semester,
                x:horizontal[matkul[i].semester], y: matkul[i].semester*150, shape:shape
            }])

            if(matkul[i].prasyarat.tempuh.length != 0){
                for (let j = 0; j < matkul[i].prasyarat.tempuh.length; j++){
                    edges.add([
                        { from: matkul[i].kode, to: matkul[i].prasyarat.tempuh[j],
                          arrows: "from"}
                    ])
                }
            }
            if(matkul[i].prasyarat.lulus.length != 0){
                for (let j = 0; j < matkul[i].prasyarat.lulus.length; j++){
                    edges.add([
                        { from: matkul[i].kode, to: matkul[i].prasyarat.lulus[j],
                          dashes: true }
                    ])
                }
            }
            if(matkul[i].prasyarat.bersamaan.length != 0){
                for (let j = 0; j < matkul[i].prasyarat.bersamaan.length; j++){
                    edges.add([
                        { from: matkul[i].kode, to: matkul[i].prasyarat.bersamaan[j],
                          arrows: "from", dashes: true }
                    ])
                }
            }
        }
    }
    fetchUsers();

    var container = document.getElementById('network');

    var data = {
```

```
            nodes: nodes,
            edges: edges
        };

        var options = {
            nodes: {
                margin: 10,
                widthConstraint: {
                    maximum: 120,
                },
            },
            physics: false,
            edges: {
                smooth: false
            },
        };

        var network = new vis.Network(container, data, options);
    }
```

– **timeline.js**, berkas *JavaScript* ini berisi fungsi khusus untuk membuat visualisai dalam bentuk *timeline*. Fungsi pada berkas ini nantinya akan dipanggil pada index.html. Untuk kodenya dapat dilihat pada Kode 4.3.

Kode 4.3: Kode *timeline.js*

```
function processTimeline(){
  document.getElementById("network").style.display="none"
  document.getElementById("timeline").style.display="inline"
  let matkul = ''
  const fetchUsers = async () => {
      try {
          const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json');
          if (!res.ok) {
              throw new Error(res.status);
          }
          const data = await res.json();
          matkul = data;
          createTimeline(matkul);
      } catch (error) {
          console.log(error);
      }
  }

  function createTimeline() {
      var groups = new vis.DataSet([
        { id: 1, content: "Mata Kuliah Wajib" },
        { id: 2, content: "Mata Kuliah Pilihan" },
      ]);

      let tahun=new Date().getFullYear();
      let bulan = new Date().getMonth();
      if(bulan<7){
        tahun-=1
        bulan=7
      }
      else{
        bulan=7
      }

      let semester=[]
      for(i=1;i<=9;i++){
          semester[i]= tahun.toString()+"-"+bulan.toString()+"- 31"
          if(i%2!=0){
            bulan+=5
          }
          else{
            bulan-=5
            tahun+=1
          }
      }

      var res = [];
      let color=["","lightBlue","yellow","red","green","pink","purple","orange","darkBlue"]
      for (var i = 0; i < matkul.length; i++) {
          let wajib=''
          if(matkul[i].wajib){
            wajib=1;
          }
          else{
            wajib=2;
          }
          res.push(
            { id: matkul[i].kode, content: matkul[i].nama, editable: false,group:wajib ,
              start: semester[matkul[i].semester],end:semester[matkul[i].semester+1],
              className: color[matkul[i].semester]  }
          )
      }
      var items = new vis.DataSet(res);

      var container = document.getElementById('timeline');

      var options = {};

      var timeline = new vis.Timeline(container, items,groups, options);
    }
```

```
71            fetchUsers();
72        }
```

  - **vis.js**, berkas *JavaScript* ini berisi fungsi - fungsi yang sudah dibuatkan oleh *Vis.js*. Berkas ini akan dipanggil pada index.html.
- **folder `node_modules`**, folder ini berfungsi untuk menyimpan *library Node.js*.
- **.gitignore**, berkas ini hanya berisikan nama folder yang tidak akan terbawa saat kita melakukan *commit* program ke *github* karena ukurannya yang terlalu besar, yaitu folder *node_modules*.
- **index.html**, berkas HTML ini merupakan berkas utama untuk menjalankan aplikasi VisKur. Nantinya berkas ini akan dipanggil pada main.js. Untuk kodenya dapat dilihat pada Kode 4.4

Kode 4.4: Kode *index.html*

```
<html>
<head>
    <script type="text/javascript" src="js/vis.js"></script>
    <link href="css/vis-network.min.css" rel="stylesheet" type="text/css" />
    <link href="css/vis-timeline-graph2d.min.css" rel="stylesheet" type="text/css" />

    <script type="text/javascript" src="js/network.js"></script>
    <script type="text/javascript" src="js/timeline.js"></script>
</head>

<body><br>
    <button type="button" onclick="processNetwork()">Network</button>
    <button type="button" onclick="processTimeline()">Timeline</button><br><br>
    <div id="network"></div>
    <div id="timeline"></div>
</body>
</html>
```

- **main.js**, berkas *JavaScript* ini berisi fungsi - fungsi yang akan dijalankan saat mengeksekusi aplikasi *Electron*. Untuk kodenya dapat dilihat pada Kode 4.5.

Kode 4.5: Kode *main.js*

```
const { app, BrowserWindow } = require('electron')
const path = require('path')

function createWindow () {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js')
    }
  })

  win.loadFile('index.html')
}

app.whenReady().then(() => {
  createWindow()
})

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit()
  }
})
```

- **package-lock.json**, berkas .json ini secara otomatis dibuatkan saat melakukan install *Node Package Manager* (npm), dimana akan memodifikasi folder ***`node_modules`***.
- **package.json**, berkas *.json* (*JavaScript Object Notation*) ini berisi tentang deskripsi dari project *Node.js*. Untuk kodenya dapat dilihat pada Kode 4.6

Kode 4.6: Kode *package.json*

```
{
  "name": "VisKur",
  "version": "1.0.0",
  "description": "Visualisasi Kurikulum",
  "main": "main.js",
  "scripts": {
    "start": "electron ."
  },
  "author": "Joshua Delavo",
  "license": "MIT",
  "devDependencies": {
    "electron": "^16.0.0"
  }
}
```

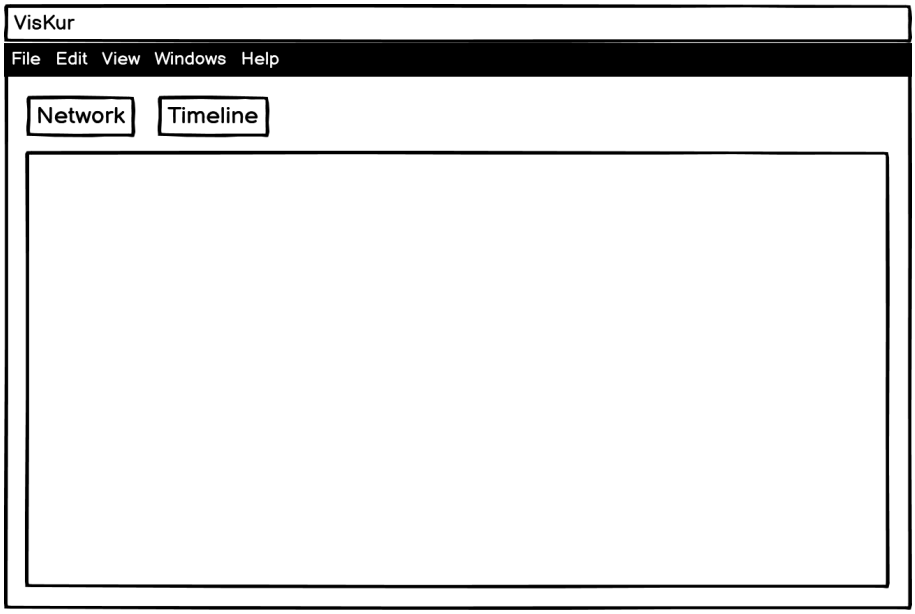
- **preload.js**, berkas *JavaScript* ini untuk mengakses *Node.js*. Skrip ini akan dijalankan sebelum proses render dimuat. Untuk kodenya dapat dilihat pada Kode 4.7

Kode 4.7: Kode *preload.js*

```
window.addEventListener('DOMContentLoaded', () => {
  const replaceText = (selector, text) => {
    const element = document.getElementById(selector)
    if (element) element.innerText = text
  }

  for (const type of ['chrome', 'node', 'electron']) {
    replaceText(`${type}-version`, process.versions[type])
  }
})
```

4.2 Perancangan Antarmuka

Perancangan antarmuka dibuat sesuai kebutuhan pengguna Aplikasi VisKur. Antarmuka ini hanya akan mengeluarkan antarmuka keluaran seperti pada gambar 4.2. Berikut ini merupakan rancangan antarmuka aplikasi yang akan dibuat:



Gambar 4.2: Rancangan Antarmuka Aplikasi VisKur

Rancangan antarmuka ini merukapan rancangan antarmuka masukan sekaligus merupakan rancangan antarmuka hasil, dimana pengguna hanya akan dapat menekan tombol *Network* ataupun tombol *Timeline*, dimana hasil visualisasinya akan langsung keluar pada halaman di bawahnya. Pengguna dapat melakukan pindah halaman visualisai hanya dengan menekan salah satu di antara kedua tombol tersebut. *Menubar* yang berisi menu *File*, *Edit*, *View*, *Windows*, dan *Help* merupakan menu bawaan dari aplikasi *Electron* yang berfungsi untuk mengoperasikan aplikasi *Electron* itu sendiri.

BAB 5

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

5.1.1 Lingkungan Implementasi

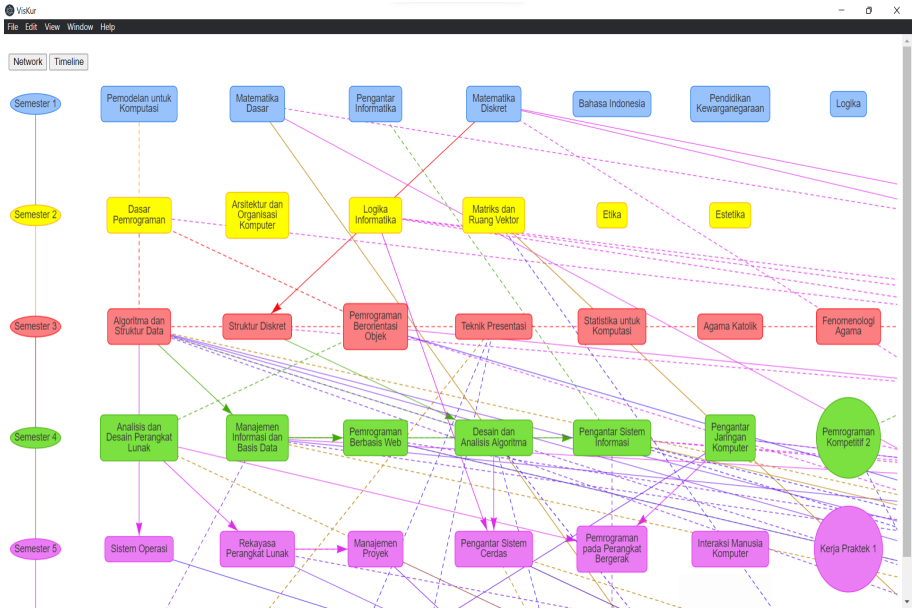
Implementasi perangkat lunak ini dilakukan di komputer penulis dengan spesifikasi sebagai berikut:

1. *Processor*: Intel Core i7-9750H
2. *Random Access Memory* (RAM): 16GB DDR4
3. *Graphics Processing Unit* (GPU): NVIDIA GeForce GTX 1650
4. *Storage*: 512 GB SSD
5. Sistem Operasi: Windows 11
6. Versi *node*: v14.17.6
7. Versi *npm*: 6.14.15

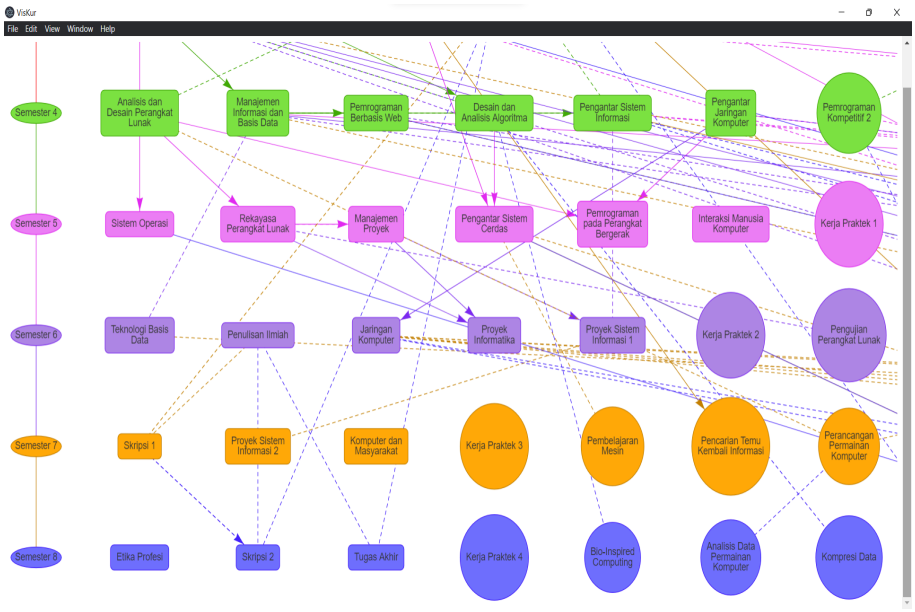
5.1.2 Hasil Implementasi

Hasil implementasi aplikasi VisKur berupa tampilan visualisasi dalam bentuk *Network* dan *Timeline*.

- Contoh hasil visualisasi *Network* dapat dilihat pada Gambar 5.1 dan Gambar 5.2.

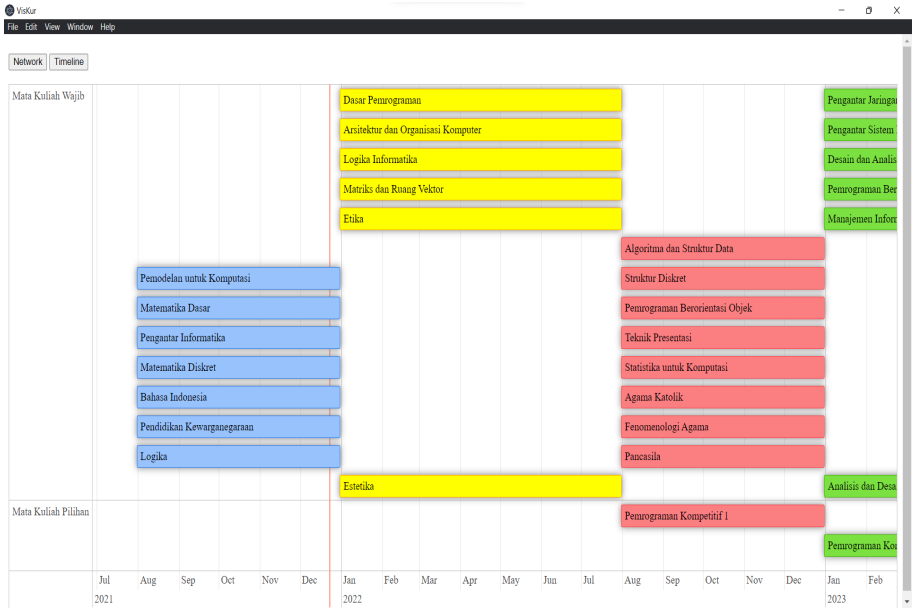


Gambar 5.1: Hasil visualisasi *Network*1

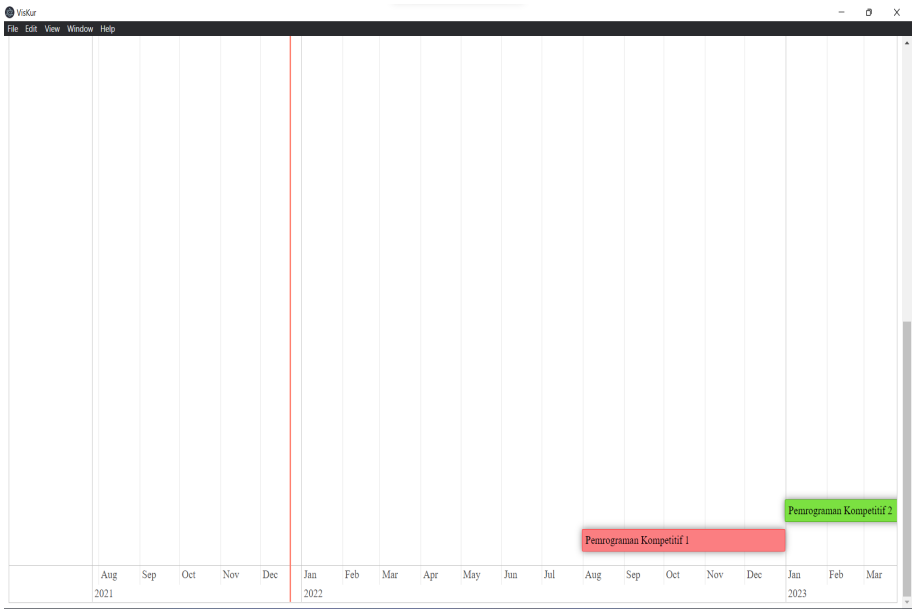


Gambar 5.2: Hasil visualisasi *Network*2

- Contoh hasil visualisasi *Timeline* dapat dilihat pada Gambar 5.3 dan Gambar 5.4.



Gambar 5.3: Hasil visualisasi *Timeline*1



Gambar 5.4: Hasil visualisasi *Timeline*2

5.2 Pengujian

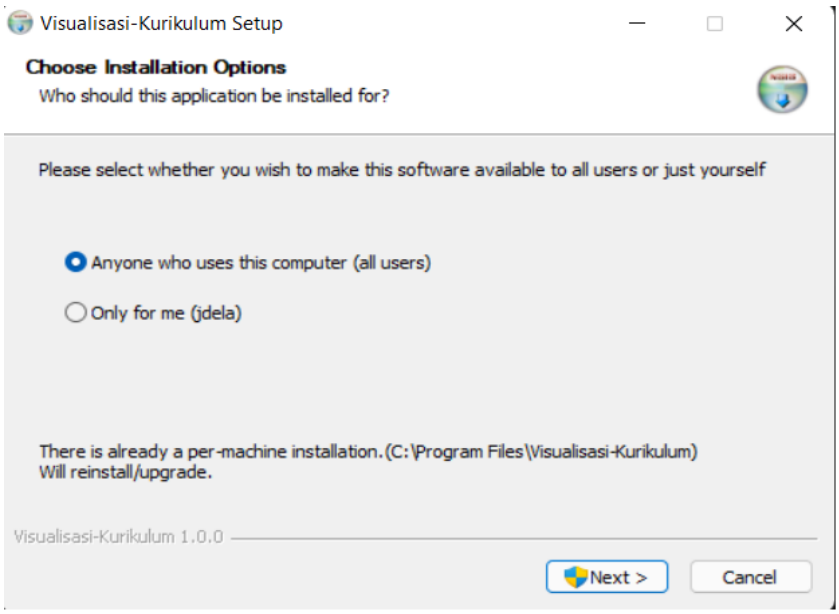
5.2.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui keberhasilan pemasangan perangkat lunak di komputer penulis dengan spesifikasi seperti berikut :

1. *Processor*: Intel Core i7-9750H
2. *Random Access Memory* (RAM): 16GB DDR4
3. *Graphics Processing Unit* (GPU): NVIDIA GeForce GTX 1650
4. Sistem Operasi: Windows 11
5. Resolusi Layar: 1920 x 1080

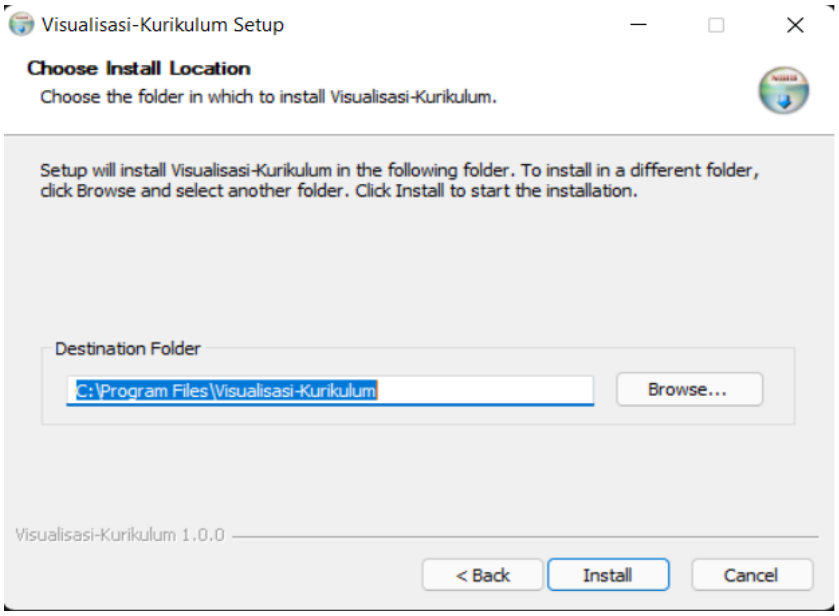
Berikut merupakan langkah - langkah yang dilakukan untuk memasang aplikasi VisKur:

- Jalankan aplikasi Visualisasi-Kurikulum Setup 1.0.0.exe, maka akan muncul seperti gambar 5.5. Kemudian pilih apakah aplikasi dapat digunakan untuk semua pengguna atau hanya untuk saya.



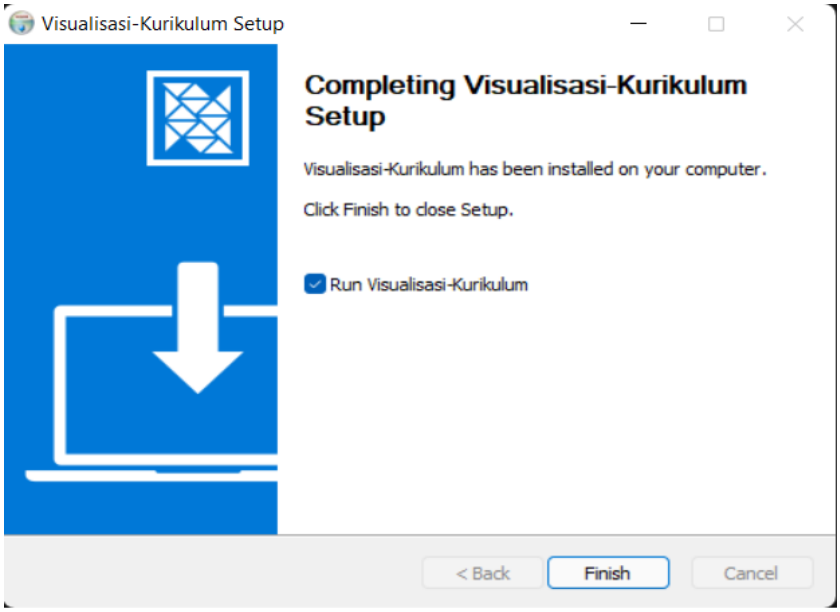
Gambar 5.5: *Setup* aplikasi VisKur

- Pilih *folder* tujuan untuk tempat penyimpanan aplikasi seperti pada Gambar 5.6, kemudian klik tombol *install* untuk memasang aplikasi.



Gambar 5.6: *Setup* aplikasi VisKur

- Setelah aplikasi berhasil dipasang, maka akan keluar seperti pada Gambar 5.7, kemudian klik tombol *Finish* dan aplikasi akan langsung dijalankan.

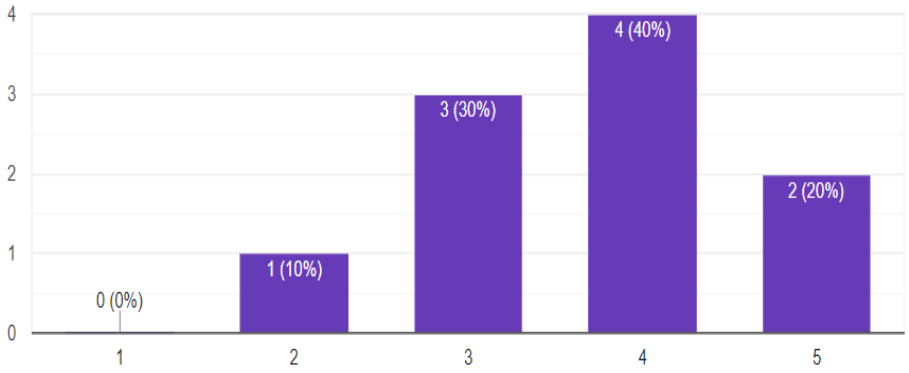


Gambar 5.7: *Setup* aplikasi VisKur

5.2.2 Pengujian Ekperimental

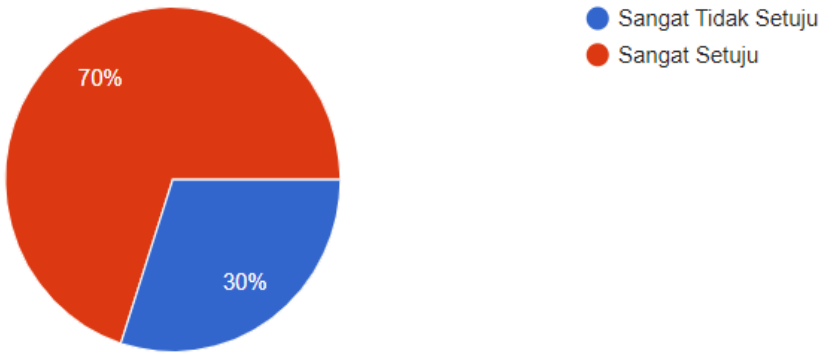
Pengujian eksperimental pada skripsi ini akan dilakukan dengan melakukan survei kepada sepuluh orang mahasiswa aktif Universitas Katolik Parahyangan jurusan informatika yang kemudian didapatkan hasil seperti berikut:

- Hasil diagram batang pada Gambar 5.8 menunjukkan bahwa sembilan orang satu orang memilih tidak setuju, tiga orang memilih biasa saja, empat orang memilih setuju, dan dua orang memilih sangat setuju bahwa lebih mudah dimengerti daripada pohon kurikulum.



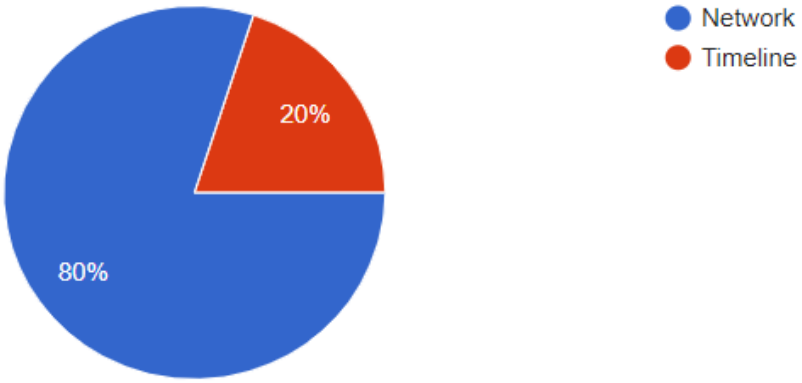
Gambar 5.8: Diagram batang hasil pemilihan aplikasi VisKur terhadap pohon kurikulum

- Hasil diagram lingkaran pada Gambar 5.9 menunjukkan bahwa tujuh puluh persen mahasiswa memilih sangat setuju dan tiga puluh persen mahasiswa memilih sangat tidak setuju bahwa informasi yang ditampilkan oleh aplikasi VisKur lebih terpakai untuk membantu memilih mata kuliah yang akan diambil pada semester berikutnya dibandingkan dengan informasi yang ditampilkan pada pohon kurikulum. Untuk informasi yang tidak ditampilkan pada aplikasi VisKur adalah jumlah sks untuk setiap mata kuliahnya, sedangkan informasi yang tidak ada pada pohon kurikulum adalah mata kuliah pilihan.



Gambar 5.9: Diagram lingkaran pemilihan aplikasi VisKur terhadap Pohon kurikulum

- Untuk kelebihan pohon kurikulum, menurut mereka adalah mata kuliah yang disajikan lebih jelas dan lengkap informasinya karena memiliki jumlah sks untuk setiap mata kuliahnya.
- Untuk kekurangan pohon kurikulum, menurut mereka adalah panah yang membingungkan karena warna dari setiap panahnya sama semua dan bertumpuk - tumpuk sehingga agak susah untuk memahami alurnya. Kemudian tidak adanya informasi mata kuliah pilihan serta desain yang kurang menarik atau tidak interaktif.
- Untuk kelebihan aplikasi VisKur, menurut mereka adalah penampilannya lebih menarik karena pemberian warna dan bentuk yang berbeda membantu mereka dalam membaca kurikulum. Alur hubungan antar mata kuliahnya lebih jelas sehingga lebih mudah untuk melihat prasyaratnya untuk setiap mata kuliah yang ada.
- Untuk kekurangan aplikasi Viskur, menurut mereka adalah informasi yang ditampilkan tidak selengakap informasi yang ditampilkan pada pohon kurikulum karena tidak ada jumlah sks untuk setiap mata kuliahnya.
- Hasil dari diagram lingkaran pada Gambar 5.10 menunjukkan bahwa delapan puluh persen mahasiswa memilih jenis visualisasi *Network* dan dua puluh persen mahasiswa memilih jenih visualisasi *Timeline* yang lebih cocok untuk memvisualisasikan kurikulum 2018.



Gambar 5.10: *Setup* aplikasi VisKur

- Alasan mereka lebih memilih bentuk visualisasi *Network* adalah karena penampilannya lebih menarik, lebih mudah dipahami, lebih terlihat hubungannya, serta lebih interaktif karena dapat digerakan sesuai dengan kebutuhan.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil dari analisis, implementasi, dan pengujian Aplikasi VisKur yang telah dibuat, telah diperoleh kesimpulan sebagai berikut:

- Berdasarkan tujuan dari skripsi ini yang berfungsi untuk membantu mahasiswa informatika Universitas Katolik Parahyangan sudah dapat berhasil terlaksanakan karena dilihat dari hasil survei yang dilakukan sebagian besar mahasiswa lebih memilih menggunakan aplikasi VisKur daripada menggunakan pohon kurikulum untuk melihat kurikulum 2018 dengan alasan yang telah disebutkan pada 5.2.2.
- Aplikasi VisKur telah dapat mengambil data dari *API* kemudian membuatkan visualisasinya dalam bentuk *Network* dan *Timeline*.
- Aplikasi VisKur telah dapat dipasang dan dijalankan pada seluruh perangkat dengna sistem operasi *windows*, baik *windows* 10 maupun *windows* 11.

6.2 Saran

Dari hasil penelitian termasuk kesimpulan yang didapat, berikut adalah saran untuk pengembang selanjutnya:

- Memperbaiki tampilan aplikasi VisKur sehingga *button Network* dan *Timeline* tidak sesederhana saat ini.
- Penambahan fitur saran pengambilan matakuliah untuk mahasiswa untuk mata kuliah di semester berikutnya secara umum.

DAFTAR REFERENSI

[1] Adithia, M. T., Nugraheni, C. E., Hakim, H., Moertini, V. S., dan Wijaya, C. (2018) *kurikulum-2018*. Universitas Katolik Parahyangan, Bandung.

[2] Electron. https://www.electronjs.org/docs/latest. Accessed: 2021-04-27.

[3] Process model electron. https://www.electronjs.org/docs/latest/tutorial/process-model. Accessed: 2021-04-27.

[4] Install electron. https://www.electronjs.org/docs/latest/tutorial/quick-start. Accessed: 2021-04-27.

[5] Vis.js. https://visjs.org/. Accessed: 2021-04-27.

[6] Timeline. https://visjs.github.io/vis-timeline/docs/timeline/. Accessed: 2021-04-27.

[7] Network. https://visjs.github.io/vis-network/docs/network/. Accessed: 2021-04-27.

[8] Dataset. https://visjs.github.io/vis-data/data/index.html. Accessed: 2021-04-27.

[9] Graph2d. https://visjs.github.io/vis-timeline/docs/graph2d/. Accessed: 2021-04-27.

[10] Graph3d. https://visjs.github.io/vis-graph3d/docs/graph3d/index.html. Accessed: 2021-04-27.

[11] Ftis open data. https://github.com/ftisunpar/data. Accessed: 2021-04-27.

[12] Javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript. Accessed: 2021-10-29.

[13] Async wait. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await. Accessed: 2021-10-29.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello␣$@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

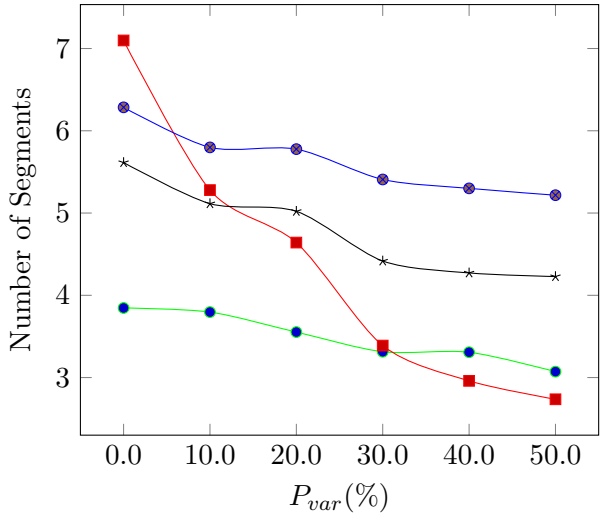
Kode A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

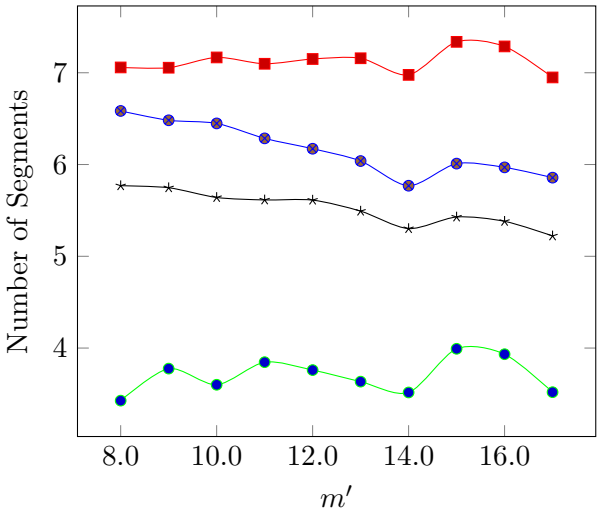
LAMPIRAN B

HASIL EKSPERIMEN

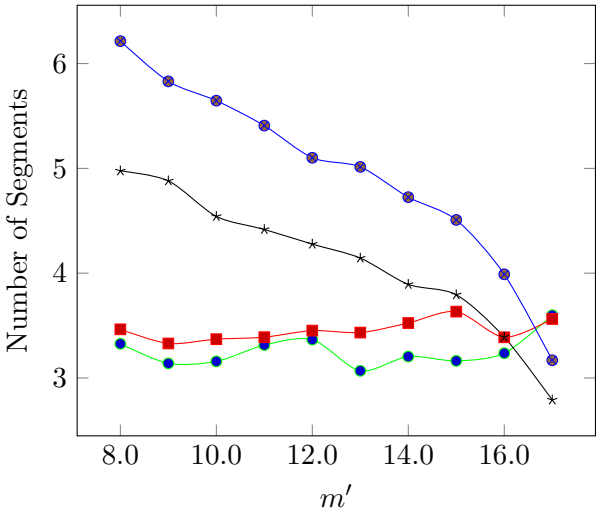
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



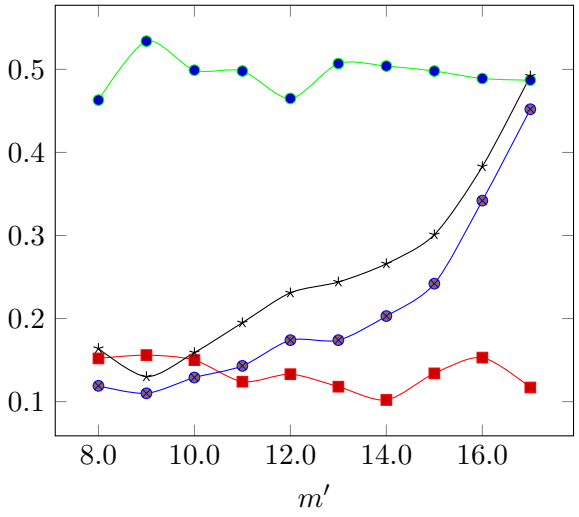
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4