

## 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 OF 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**

**Pascal Alfadian, Nugroho, M.Comp.**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**Vania Natali, S.Kom.**

**Raymond Chandra Putra, S.T., M.T.**

**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 dijatuhan 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 Teknik 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 dan juga saling bertumpuk.

Aplikasi VisKur akan mengambil data dari *API* milik Fakultas Teknologi Infromasi dan Sains (FTIS) Unpar kemudian memvisualisasikannya dalam bentuk *Network* dan *Timeline* menggunakan *framework Electron* dengan *library Vis.js*. VisKur dapat berjalan pada komputer dengan sistem operasi *Windows*, baik *Windows 10* maupun *Windows 11*.

Setelah dilakukan pengujian terhadap aplikasi VisKur dengan melakukan survei kepada beberapa mahasiswa aktif Unpar jurusan Teknik Informatika, didapatkan hasil bahwa aplikasi VisKur dapat menyajikan kurikulum 2018 dengan lebih baik dibanding dengan pohon kurikulum yang terdapat pada juklak.

**Kata-kata kunci:** VisKur, Electron, Vis.js, juklak, Teknik Informatika, Unpar



## ABSTRACT

Every single Parahyangan Catholic University student majoring in Informatics Engineering needs to fill out a study plan form (FRS) to take courses in the next semester. To be able to find out what courses can be taken in the next semester, students must look at the curriculum tree contained in the manual for implementing academic activities (juklak). However, the curriculum tree has drawbacks in that there are no elective courses and it is difficult to see the prerequisites for each course, because the depiction of the prerequisite lines in the curriculum tree has the same color and also overlaps each other.

The VisKur application will retrieve data from the Fakultas Teknologi Infromasi dan Sains (FTIS) Unpar API and then visualize it in the form of a Network and Timeline using the Electron framework with the Vis.js library. VisKur can run on computers with Windows operating systems, both on Windows 10 or Windows 11.

After testing the VisKur application by conducting a survey of several active Unpar students majoring in Informatics Engineering, it was found that the VisKur application was able to present the 2018 curriculum better than the curriculum tree contained in the operational guidelines.

**Keywords:** VisKur, Electron, Vis.js, juklak, Informatics, Unpar



*Untuk orang tua yang telah membayai 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 sehingga 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 Natali, S.Kom., M.T. selaku dosen pengaji yang telah memberikan kritik dan saran kepada penulis.
- Bapak Raymond Chandra Putra, S.T., M.T. selaku dosen pengaji 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

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Formulir Rencana Studi . . . . .	5
2.2 Kurikulum 2018 . . . . .	5
2.2.1 Kodifikasi . . . . .	5
2.2.2 Bobot Pemrograman . . . . .	6
2.2.3 Prasyarat Mata Kuliah . . . . .	7
2.3 Electron . . . . .	7
2.4 Vis.js . . . . .	11
2.4.1 Timeline . . . . .	12
2.4.2 Network . . . . .	12
2.4.3 DataSet . . . . .	13
2.4.4 Graph2d . . . . .	13
2.4.5 Graph3d . . . . .	14
2.5 FTIS Open Data . . . . .	14
2.6 JavaScript . . . . .	15
2.7 Async and Await . . . . .	16
<b>3 ANALISIS</b>	<b>17</b>
3.1 Analisis Bentuk Data . . . . .	17
3.2 Analisis Bentuk Visualisasi . . . . .	18
3.2.1 Pohon kurikulum 2018 . . . . .	18
3.2.2 Graph2d . . . . .	18
3.2.3 Graph3d . . . . .	18
3.2.4 Timeline . . . . .	19
3.2.5 Network . . . . .	19
3.3 Analisis Sistem Visualisasi . . . . .	19
3.3.1 Spesifikasi Kebutuhan Perangkat Lunak . . . . .	19
3.3.2 Use Case Diagram . . . . .	19
3.3.3 Flow Chart Diagram . . . . .	20

3.3.4	Perancangan Modul . . . . .	21
<b>4</b>	<b>PERANCANGAN</b>	<b>25</b>
4.1	Perancangan Struktur Aplikasi . . . . .	25
4.2	Perancangan Antarmuka . . . . .	29
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>31</b>
5.1	Implementasi . . . . .	31
5.2	Pengujian . . . . .	33
5.2.1	Lingkungan Pengujian . . . . .	33
5.2.2	Pengujian Fungsional . . . . .	33
5.2.3	Pengujian Ekperimental . . . . .	37
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>41</b>
6.1	Kesimpulan . . . . .	41
6.2	Saran . . . . .	41
<b>DAFTAR REFERENSI</b>		<b>43</b>
<b>A</b>	<b>KODE PROGRAM</b>	<b>45</b>
<b>B</b>	<b>HASIL EKSPERIMEN</b>	<b>49</b>

## DAFTAR GAMBAR

2.1	Kodifikasi mata kuliah . . . . .	6
2.2	Rincian Bobot Pemrograman . . . . .	6
2.3	Daftar mata kuliah wajib semester empat beserta prasyaratnya . . . . .	7
2.4	Daftar mata kuliah pilihan beserta prasyaratnya (1) . . . . .	7
2.5	Proses <i>render Chrome</i> . . . . .	8
2.6	Contoh membuat program <i>Electron</i> . . . . .	11
2.7	Hasil contoh untuk membuat <i>timeline</i> menggunakan <i>vis.js</i> . . . . .	12
2.8	Hasil contoh <i>timeline</i> setelah digeser ke kanan. . . . .	12
2.9	Hasil contoh <i>timeline</i> setelah digulir ke atas. . . . .	12
2.10	Hasil contoh untuk membuat <i>network</i> menggunakan <i>vis.js</i> . . . . .	13
2.11	Hasil contoh untuk membuat <i>graph2d</i> menggunakan <i>vis.js</i> . . . . .	14
2.12	Hasil contoh untuk membuat <i>graph3d</i> menggunakan <i>vis.js</i> . . . . .	14
3.1	Contoh data kurikulum pada <i>API</i> . . . . .	17
3.2	Pohon kurikulum 2018 yang terdapat pada juklak . . . . .	18
3.3	Use Case Diagram . . . . .	19
3.4	<i>Flowchart</i> aplikasi VisKur . . . . .	21
4.1	Rancangan Antarmuka Aplikasi VisKur . . . . .	30
5.1	Hasil visualisasi <i>Network</i> secara keseluruhan . . . . .	31
5.2	Hasil visualisasi <i>Network</i> secara lebih jelas . . . . .	32
5.3	Hasil visualisasi <i>Timeline</i> secara keseluruhan . . . . .	32
5.4	Hasil visualisasi <i>Timeline</i> secara lebih jelas . . . . .	33
5.5	<i>Setup</i> aplikasi VisKur . . . . .	34
5.6	<i>Setup</i> aplikasi VisKur . . . . .	34
5.7	<i>Setup</i> aplikasi VisKur . . . . .	35
5.8	Halaman utama aplikasi VisKur. . . . .	35
5.9	<i>Node</i> skripsi 1 sebelum digeser . . . . .	36
5.10	<i>Node</i> skripsi 1 setelah ditekan dan digeser ke kiri . . . . .	36
5.11	<i>Timeline</i> sebelum digulir dan digeser . . . . .	37
5.12	<i>Timeline</i> setelah digulir dan digeser . . . . .	37
5.13	Diagram batang hasil pemilihan aplikasi VisKur terhadap pohon kurikulum . . . . .	38
5.14	Diagram lingkaran pemilihan aplikasi VisKur terhadap Pohon kurikulum . . . . .	38
5.15	Diagram lingkaran pemilihan bentuk <i>Network</i> dengan bentuk <i>Timeline</i> . . . . .	39
B.1	Hasil visualisasi <i>Network</i> secara keseluruhan . . . . .	49
B.2	Hasil visualisasi <i>Network</i> secara lebih jelas . . . . .	49
B.3	Hasil visualisasi <i>Timeline</i> secara keseluruhan . . . . .	50
B.4	Hasil visualisasi <i>Timeline</i> secara lebih jelas . . . . .	50



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Saat mahasiswa jurusan Teknik Informatika Universitas Katolik Parahyangan akan melakukan pengisian formulir rencana studi (FRS), seringkali mereka kesulitan untuk membaca pohon kurikulum yang terdapat pada petunjuk pelaksanaan kegiatan akademik (juklak). Dikarenakan pada pohon kurikulum tersebut tidak terdapat mata kuliah pilihan serta sulit untuk melihat prasyarat pada setiap mata kuliah, karena penggambaran garis prasyarat pada pohon kurikulum memiliki warna yang sama dan juga saling bertumpuk. Pada setiap kurikulum juga memiliki aturan yang berbeda dalam pengambilan matakuliah ataupun matakuliah yang disediakan, 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 mengambil data kurikulum 2018 dari *API* milik *FTIS Unpar* yang kemudian akan divisualisasikan ke dalam bentuk yang cocok untuk membaca data tersebut. Data yang tersedia di *API* merupakan data kurikulum 2018 sehingga aplikasi hanya dapat memvisualisasikan data kurikulum 2018, namun jika terdapat data kurikulum lain dengan struktur yang sama, maka aplikasi ini dapat memvisualisasikannya juga. Aplikasi visualisasi ini akan dinamakan aplikasi VisKur. Dengan aplikasi ini diharapkan dapat membantu mahasiswa untuk melihat prasyarat untuk setiap mata kuliahnya melalui penggambaran garis yang lebih jelas serta terdapat mata kuliah pilihan yang secara umum diadakan pada semester - semester tertentu. Aplikasi ini juga dapat dengan mudah dipasang dan dijalankan pada komputer dengan sistem operasi *Windows*.

Menurut Kamus Besar Bahasa Indonesia (KBBI), visualisasi adalah pengungkapan gagasan atau perasaan dengan menggunakan bentuk gambar, tulisan (kata dan angka), peta, grafik, dan sebagainya. Visualisasi akan mengubah kumpulan data (data mentah) menjadi bentuk visual yang lebih menarik dan lebih mudah dipahami oleh pembaca.

*Electron* adalah salah satu kerangka kerja 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*.

## 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 cara membaca kurikulum 2018 FTIS UNPAR dari *github*?
3. Bagaimana mengimplementasikan visualisasi tersebut dalam *Vis.js*?

## 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. Perangkat lunak ini hanya untuk membantu mahasiswa Teknik Informatika Unpar saja.
2. Karena keterbatasan mesin yang dimiliki, maka hanya dibuat dalam versi *Windows* saja.

## 1.5 Metodologi

Bagian - bagian pelaksanaan 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. Membuat aplikasi VisKur.
10. Melakukan pengujian dan eksperimen.
11. 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 *Electron* 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 formulir rencana studi, kurikulum 2018, *Electron*, *Vis.js*, *FTIS Open Data*, dan *JavaScript*.

#### 2.1 Formulir Rencana Studi

Formulir rencana studi merupakan formulir yang wajib diisi oleh mahasiswa untuk pengambilan mata kuliah di semester selanjutnya. Mahasiswa Universitas Katolik Parahyangan wajib melakukan pengisian formulir rencana studi (FRS) secara *online* pada *student portal* beberapa minggu sebelum pembeleajaran semester baru dimulai. Berikut merupakan langkah - langkah yang harus dilakukan mahasiswa saat melakukan pengisian FRS:

1. Membuka student portal milik Unpar <https://studentportal.unpar.ac.id/>.
2. Lakukan *log in* dengan memasukkan alamat *email* Unpar, kemudian memasukkan password.
3. Pada halaman utama *student portal*, pilih menu FRS/PRS.
4. Pilih mata kuliah yang akan diambil untuk semester berikutnya. Untuk mahasiswa dengan nilai indeks prestasi semester (IPS) lebih besar dari tiga, dapat mengambil maksimal 24 sks. Untuk mahasiswa dengan nilai IPS antara dua setengah sampai tiga, dapat mengambil maksimal 21 sks. Untuk mahasiswa dengan nilai IPS di bawah dua setengah, hanya dapat mengambil maksimal 18 sks.
5. Jika sudah mahasiswa harus memberitahukan kepada dosen wali bahwa pengisian FRS sudah dilakukan.
6. Setelah disetujui oleh dosen wali, maka mahasiswa akan mendapat hasil registrasi FRS dari *registrasi@unpar.ac.id*, kemudian mahasiswa harus meneruskan *email* tersebut ke *informatika@unpar.ac.id*.

#### 2.2 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]

##### 2.2.1 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 digit, dengan pembacaan kode dari kiri ke kanan secara berurutan (AAABCDEEFF), yang terdiri dari:

- AAA (3 digit) - kode khas Program Studi: AIF
- BB (2 digit) - tahun diberlakukannya kurikulum (2 digit terakhir): 18
- C (1 digit) - urutan tahun pengajaran
- D (1 digit) - nomor urut KBI pengampu mata kuliah

- EE (2 digit) - nomor urut mata kuliah per semester, dengan angka pada digit terakhir sebagai penentu semester; ganjil atau genap
- FF (2 digit) - jumlah sks mata kuliah

Informasi lengkap terkait kodifikasi ini diberikan di Gambar 2.1.

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 digit terakhir sebagai penentu semester; ganjil atau genap
Jumlah sks	**	Jumlah sks

\*\*Kode mata kuliah MKU ditentukan oleh universitas

Gambar 2.1: Kodifikasi mata kuliah

### 2.2.2 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 2.2.

Bobot	Deskripsi
0.25	<ul style="list-style-type: none"> <li>Minimal 1 tugas berbentuk pembangunan program komputer</li> <li>Kuliah tidak berpraktikum</li> </ul>
0.5	<ul style="list-style-type: none"> <li>Minimal setengah dari tugas yang diberikan berbentuk pembangunan program komputer</li> <li>Kuliah tidak berpraktikum atau yang berfokus pada analisis</li> </ul>
0.75	<ul style="list-style-type: none"> <li>Di luar tugas praktikum, ada tugas kuliah berupa pembangunan program komputer</li> <li>Kuliah berpraktikum atau merupakan kuliah skripsi</li> </ul>
1	Kuliah berpraktikum dengan capaian pembelajaran adalah keahlian pemrograman atau merupakan kuliah proyek

Gambar 2.2: Rincian Bobot Pemrograman

### 2.2.3 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. Contoh rincian prasyarat mata kuliah wajib untuk semester empat dapat dilihat pada Gambar 2.3, sedangkan contoh rincian prasyarat mata kuliah pilihan dapat dilihat pada Gambar 2.4.

No	Kode	Mata Kuliah	Mata Kuliah Prasyarat	
			Tempuh	Lulus
<b>Semester 4</b>				
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 2.3: Daftar mata kuliah wajib semester empat 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 2.4: Daftar mata kuliah pilihan beserta prasyaratnya (1)

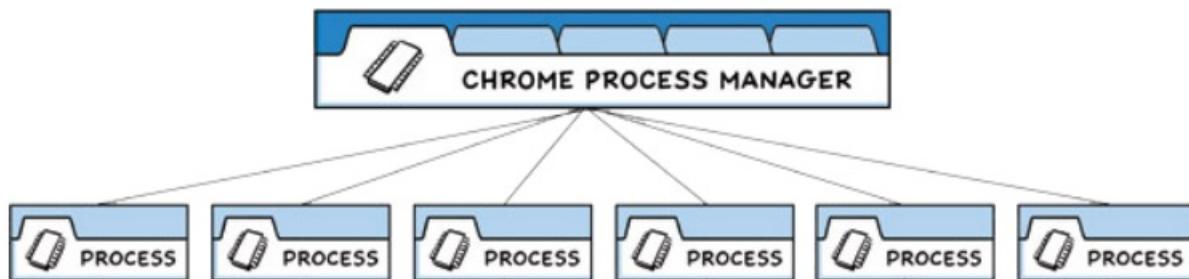
## 2.3 Electron

*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* memiliki arsitektur *multi-process* dari *Chromium*, yang membuat kerangka kerja secara arsitektur sangat mirip dengan *browser web* modern. 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. [2]

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. Untuk visualisasi proses rendernya dapat dilihat pada Gambar 2.5.



Gambar 2.5: Proses *render Chrome*

Aplikasi *Electron* terstruktur sangat mirip, dimana kita sebagai pengembang aplikasi mengontrol dua jenis proses : *main* dan *renderer*. Ini adalah analog dengan browser *Chrome* dan proses *renderer*. 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

```

1 app.on('window-all-closed', function(){
2   if(process.platform != 'darwin') app.quit()
3 })

```

Setiap aplikasi *Electron* memunculkan proses penyaji terpisah untuk setiap *BrowserWindow* yang terbuka. 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 : [2]

### 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

```

1   {
2     "name": "my-electron-app",
3     "version": "1.0.0",
4     "description": "HelloWorld!",
5     "main": "main.js",
6     "author": "JaneDoe",
7     "license": "MIT",
8   }

```

- 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

```

1   {
2     "scripts": {
3       "start": "electron."
4     }
5   }

```

- 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`

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self' '>
6      <meta http-equiv="X-Content-Security-Policy" content="default-src 'self'; script-src 'self' '>
7      <title>Hello World!</title>

```

```

8   </head>
9   <body>
10    <h1>Hello World!</h1>
11    We are using Node.js <span id="node-version"></span>,
12    Chromium <span id="chrome-version"></span>,
13    and Electron <span id="electron-version"></span>.
14   </body>
15 </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 *CommonJS*

```
1 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

```

1 const createWindow = () => {
2   const win = new BrowserWindow({
3     width: 800,
4     height: 600
5   })
6
7   win.loadFile('index.html')
8 }

```

- 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 *Promisenya* dengan menambahkan Kode 2.8.

Kode 2.8: Memanggil fungsi *createWindow*

```
1 app.whenReady().then(() => {
2   createWindow()
3 })
```

## 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*

```
1 app.on('window-all-closed', () => {
2   if (process.platform !== 'darwin') app.quit()
3 })
```

## 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*

```

1 window.addEventListener('DOMContentLoaded', () => {
2   const replaceText = (selector, text) => {
3     const element = document.getElementById(selector)
4     if (element) element.innerText = text
5   }
6
7   for (const dependency of ['chrome', 'node', 'electron']) {
8     replaceText(`${dependency}-version`, process.versions[dependency])
9   }
10 })

```

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`

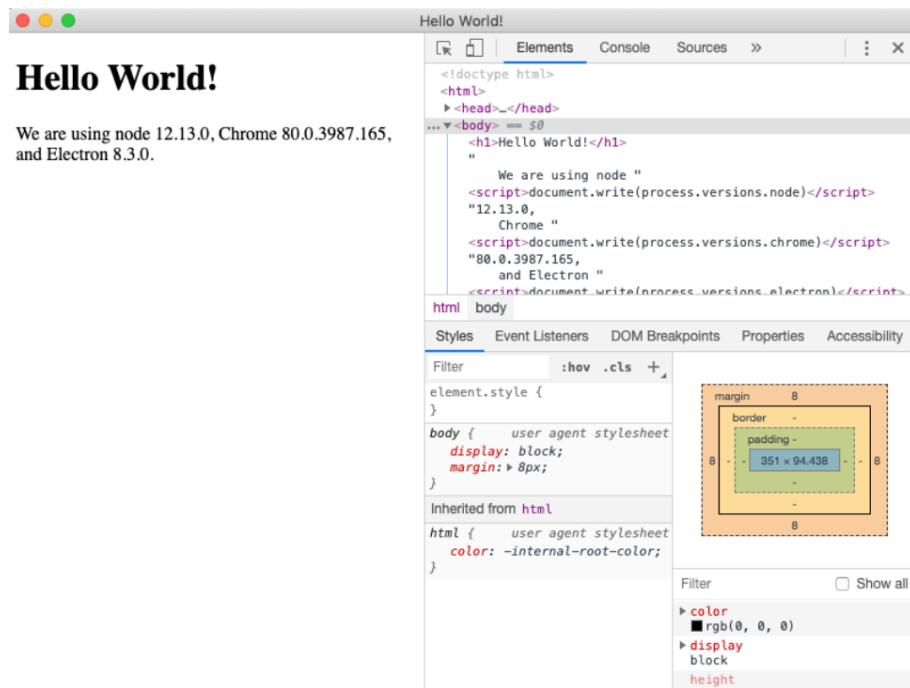
```

1 // include the Node.js 'path' module at the top of your file
2 const path = require('path')
3
4 // modify your existing createWindow() function
5 const createWindow = () => {
6   const win = new BrowserWindow({
7     width: 800,
8     height: 600,
9     webPreferences: {
10       preload: path.join(__dirname, 'preload.js')
11     }
12   })
13
14   win.loadFile('index.html')
15 }
16 // ...

```

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 2.6



Gambar 2.6: Contoh membuat program *Electron*

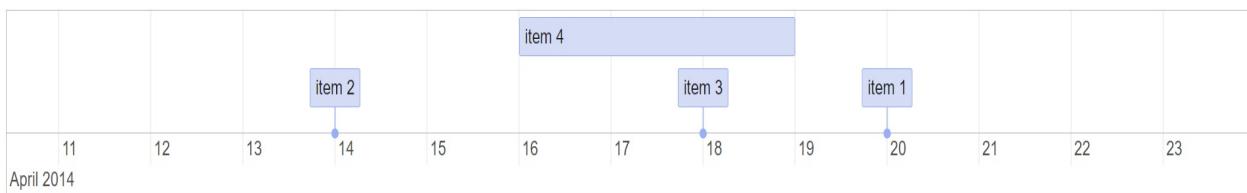
## 2.4 Vis.js

*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*. [3]

### 2.4.1 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 menggeser seperti pada Gambar 2.8 dan menggulir di *timeline* seperti pada Gambar 2.9. 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*. [3]

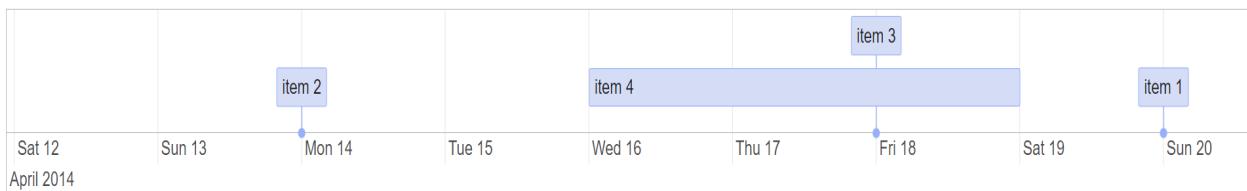
Untuk menggunakan *timeline*, kita harus menyertakan file *vis.js* dan *vis-timeline-graph2d.min.css* yang dapat diunduh dari visjs.org. Contoh hasil visualisasi dalam bentuk *timeline* dapat dilihat pada Gambar 2.7.



Gambar 2.7: Hasil contoh untuk membuat *timeline* menggunakan *vis.js*



Gambar 2.8: Hasil contoh *timeline* setelah digeser ke kanan.



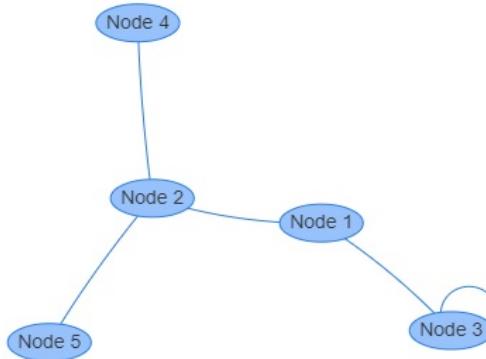
Gambar 2.9: Hasil contoh *timeline* setelah digulir ke atas.

### 2.4.2 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*. [3]

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 hasil visualisasi dalam bentuk *network* dapat dilihat pada Gambar 2.10.



Gambar 2.10: Hasil contoh untuk membuat *network* menggunakan *vis.js*

#### 2.4.3 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, dan dihapus. *DataSet* juga dapat digunakan untuk menyimpan objek *JSON* berdasarkan idnya. Objek dapat ditambahkan, diperbarui, dan dihapus dari *DataSet*. Data dalam *DataSet* dapat difilter dan diurutkan, serta *fields* (seperti tanggal) dapat dikonversi ke dalam tipe tertentu. Data dapat dinormalisasi saat menambahkannya kedalam *DataSet*. [3]

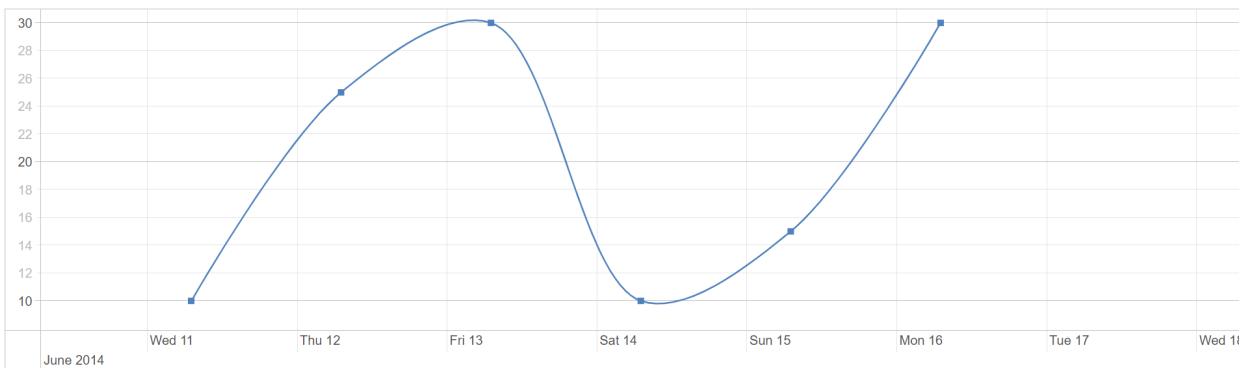
Terdapat beberapa *method* yang telah disediakan, seperti :

- *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.

#### 2.4.4 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*. [3]

Contoh hasil visualisasi dalam bentuk *graph2d* dapat dilihat pada gambar 2.11.

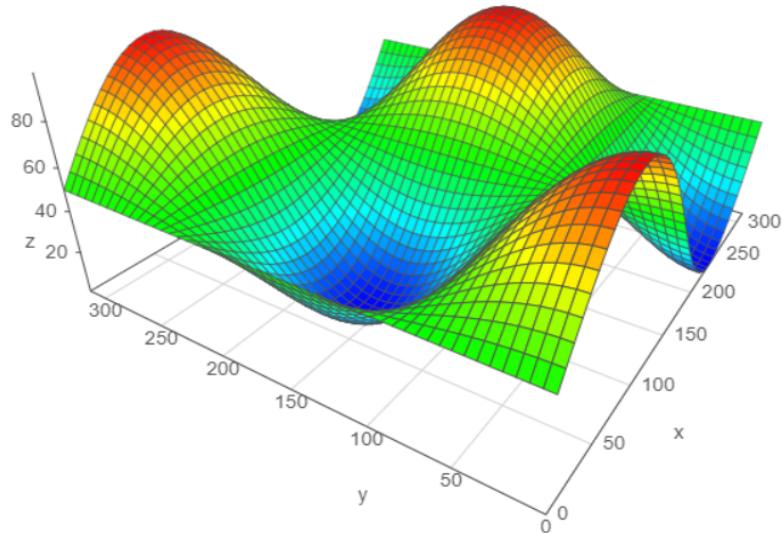


Gambar 2.11: Hasil contoh untuk membuat *graph2d* menggunakan *vis.js*

#### 2.4.5 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 meng gulir 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. [3]

Contoh hasil visualisasi dalam bentuk *graph3d* dapat dilihat pada gambar 2.12.



Gambar 2.12: Hasil contoh untuk membuat *graph3d* menggunakan *vis.js*

### 2.5 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>. [4]

*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 mata kuliah, jumlah SKS, posisi semester, serta prasyarat yang ada. Dengan bentuk datanya adalah array untuk setiap mata kuliahnya. Setiap struktur mata kuliah 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 2.12.

Kode 2.12: prasyarat.json

```

1 {
2   "kode": "AIF181100",
3   "nama": "Dasar Pemrograman",
4   "prasyarat": {
5     "tempuh": [],
6     "lulus": [
7       "AIF181101"
8     ],
9     "bersamaan": [],
10    "berlakuAngkatan" : 2018
11  },
12  "sks": 4,
13  "wajib": true,
14  "semester": 2
15 }
```

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 mata kuliah ini mulai berlaku semenjak angkatan x.

## 2.6 JavaScript

*JavaScript* (sering disingkat menjadi JS) adalah sebuah bahasa yang mendukung gaya pemrograman berorientasi objek yang dikenal sebagai bahasa *scripting* untuk halaman web. *JavaScript* merupakan bahasa yang dinamis, imperatif, dan fungsional. [5]

*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.

*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.

## 2.7 Async and Await

Penambahan fungsi pada bahasa *JavaScript* adalah fungsi *async* dan kata kunci *await*. Fitur-fitur ini membuat kode *asinkron* lebih mudah untuk ditulis dan dibaca. [6]

Fungsi *async* adalah fungsi yang mengetahui bagaimana mengharapkan kemungkinan kata kunci *await* digunakan untuk memanggil *asynchronous code*. Nilai pengembalinya 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 memvisualisasikan kurikulum 2018, diperlukan sumber data kurikulum 2018 yang terdapat pada github API. Contoh data kurikulum yang terdapat pada API dapat dilihat pada Gambar 3.1.

```
{  
    "kode": "AIF181101",  
    "nama": "Pemodelan untuk Komputasi",  
    "prasyarat": {  
        "tempuh": [],  
        "lulus": [],  
        "bersamaan": [],  
        "berlakuAngkatan" : null  
    },  
    "skls": 3,  
    "wajib": true,  
    "semester": 1  
},
```

Gambar 3.1: Contoh data kurikulum pada *API*

Untuk pengambilan datanya terdapat pada variable const fetchUsers dimana terdapat fungsi *asynchronous* dengan kata kunci await untuk memanggil fungsi API web dengan fungsi *fetch*. Bentuk datanya berupa array untuk setiap mata kuliahnya. Setiap struktur mata kuliahnya memiliki properti, sebagai berikut:

- kode: berisi kode mata kuliah.
- nama: berisi nama mata kuliah.
- prasyarat:
  - tempuh: berisi kode mata kuliah sebagai syarat tempuh.
  - lulus: berisi kode mata kuliah sebagai syarat lulus.
  - bersamaan: berisi kode mata kuliah sebagai mata kuliah yang harus diambil secara bersamaan.
  - berlakuAngkatan: berisi tahun dan akan berisi null jika tidak ada.
- sks: berisi jumlah sks dalam bentuk *integer*.
- wajib: berisi *true* untuk mata kuliah wajib dan *false* untuk mata kuliah pilihan.
- semester: berisi semester dari mata kuliahnya dalam bentuk *integer*.

## 3.2 Analisis Bentuk Visualisasi

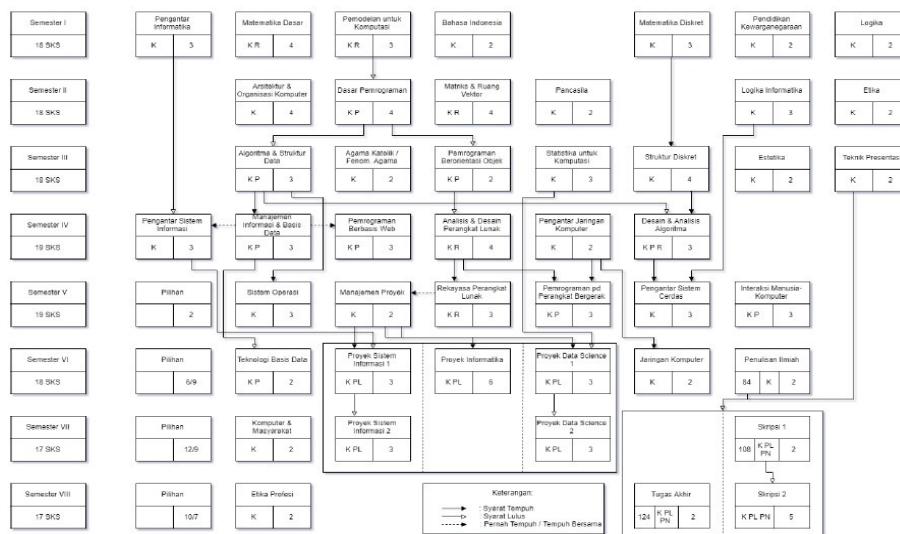
Pada subbab ini akan dijelaskan mengenai pohon kurikulum 2018 yang terdapat pada juklak kemandian akan dijelaskan mengenai jenis visualisasi apa yang dapat digunakan untuk memvisualisasikan kurikulum 2018 dengan Vis.js, dimana *Vis.js* memiliki 4 buah jenis visualisasi, yaitu *graph2d*, *graph3d*, *timeline*, dan *network*.

### 3.2.1 Pohon kurikulum 2018

Pohon kurikulum 2018 yang terdapat pada juklak memiliki bentuk visualisasi dalam bentuk *tree* yang dapat dilihat pada Gambar 3.2. Dalam pohon ini terdapat informasi nama mata kuliah, jumlah SKS, dan bentuk pembelajaran untuk setiap mata kuliahnya. Terdapat juga syarat minimal SKS wajib lulus di beberapa mata kuliah. Terdapat tiga jenis garis yang melambangkan ketiga jenis prasyarat, yaitu:

1. Garis dengan ujung anak panah terisi penuh yang melambangkan syarat tempuh.
2. Garis dengan ujung anak panah kosong yang melambangkan syarat lulus.
3. Garis putus - putus dengan ujung anak penuh terisi penuh yang melambangkan pernah tempuh atau tempuh bersama

**POHON KURIKULUM 2018 TEKNIK INFORMATIKA UNPAR**



Gambar 3.2: Pohon kurikulum 2018 yang terdapat pada juklak

### 3.2.2 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.

### 3.2.3 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.

### 3.2.4 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.

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. Diasumsikan untuk semester ganjil akan dimulai dari bulan Agustus sampai bulan Desember dan untuk semester genap akan dimulai dari bulan Januari sampai bulan Juli dikarenakan jadwal sebenarnya yang berubah - ubah akibat pandemi. Maka dari itu, besar *content* akan mengikuti waktu untuk setiap semesternya.

### 3.2.5 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*.

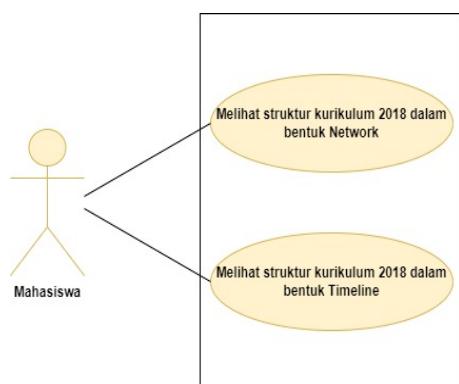
*Node* juga akan berbentuk kotak untuk mata kuliah wajib dan berbentuk lingkaran untuk mata kuliah pilihan. Untuk matakuliah a yang memiliki prasyarat b dapat dimodelkan dengan *edges* berarah dari *node* b ke *node* a. Terdapat delapan buah node tambahan yang berisikan semester satu sampai semester 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*. *Input* perangkat lunak ini sesuai dengan keinginan pengguna, apakah ingin dibuatkan visualisasi dalam bentuk *network* atau *timeline*. Kemudian untuk *outputnya* perangkat lunak akan mengambil data kurikulum 2018 dari *API*, kemudian akan menampilkan bentuk visualisasi sesuai *input* dari pengguna.

### 3.3.2 Use Case Diagram



Gambar 3.3: Use Case Diagram

Pada diagram *use case* (gambar : 3.3) terdapat dua buah fitur bagi mahasiswa yaitu, melihat struktur kurikulum 2018 dalam bentuk *Timeline* dan melihat struktur kurikulum 2018 dalam bentuk *Network*.

#### **Scenario Use Case**

##### 1. Melihat struktur kurikulum 2018 dalam bentuk *Timeline*.

- Nama: Melihat struktur kurikulum 2018 dalam bentuk *Timeline*.
- Aktor: Makasiswa
- Kondisi awal: Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk *Timeline*.
- Kondisi akhir: Aplikasi menampilkan hasil visualisasi dalam bentuk *Timeline*.
- Deskripsi: Mahasiswa menekan tombol *Timeline* pada aplikasi VisKur.
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk Timeline	Sistem akan megambil data kurikulum 2018 dari API
2	Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk Timeline	Sistem akan memvisualisasikan data yang telah diambil ke dalam bentuk Timeline

##### 2. Melihat struktur kurikulum 2018 dalam bentuk *Network*.

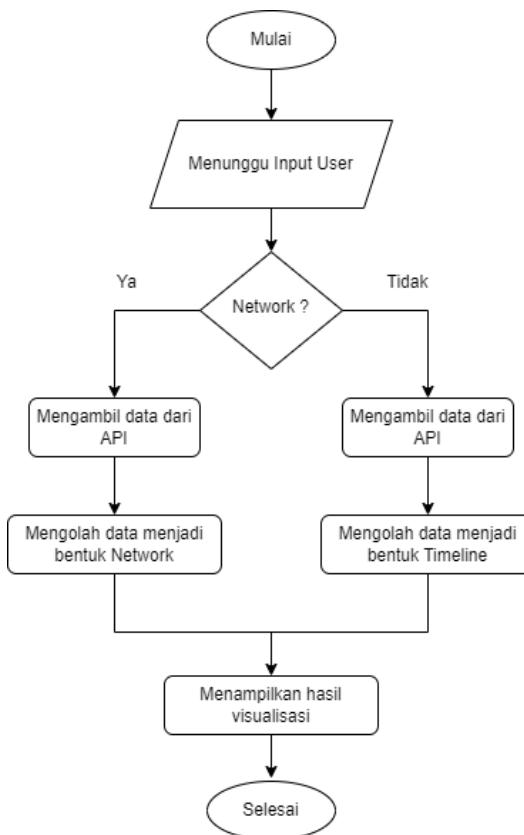
- Nama: Melihat struktur kurikulum 2018 dalam bentuk *Network*.
- Aktor: Makasiswa
- Kondisi awal: Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk *Network*.
- Kondisi akhir: Aplikasi menampilkan hasil visualisasi dalam bentuk *Network*.
- Deskripsi: Mahasiswa menekan tombol *Network* pada aplikasi VisKur.
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk Network	Sistem akan megambil data kurikulum 2018 dari API
2	Mahasiswa ingin melihat struktur kurikulum 2018 dalam bentuk Network	Sistem akan memvisualisasikan data yang telah diambil ke dalam bentuk Network

#### **3.3.3 Flow Chart Diagram**

Untuk diagaram alur aplikasi VisKur dapat dilihat pada Gambar 3.4.

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



Gambar 3.4: Flowchart aplikasi VisKur

### 3.3.4 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, dimana dapat dilihat pada baris empat.
- Untuk kodennya dapat dilihat pada Kode 3.1.

Kode 3.1: Kode pengambilan data dari API

```

1 let matkul = '';
2 const fetchUsers = async () => {
3     try {
4         const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat
5             .json');
6         if (!res.ok) {
7             throw new Error(res.status);
8         }
9         const data = await res.json();
10        matkul = data;
11        createNetwork();
12    } catch (error) {
13        console.log(error);
14    }
  
```

- 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, dimana dapat dilihat baris enam sampai baris dua belas.

Selanjutnya terdapat sebuah *looping* untuk membuat *node - node* sesuai dengan jumlah mata kuliah yang terdapat pada baris empat belas sampai dua puluh tujuh, dimana untuk bentuk dan posisi *nodenya* akan diatur di dalamnya.

Pada *looping* ini juga akan dibuatkan *edges* untuk setiap *nodenya* sesuai dengan prasyaratnya masing - masing yang terdapat pada baris dua puluh sembilan sampai baris lima puluh tiga, dimana untuk bentuk *edgesnya* 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 kodennya dapat dilihat pada Kode 3.2.

Kode 3.2: Kode untuk membuat Visualisasi Network

```

1  const horizontal = [];
2  for (i = 1; i < 9; i++) {
3      nodes.add([
4          id: i, label: "Semester" + i, group: i, y: i*150
5      ])
6      if (i != 8) {
7          edges.add([
8              { from: i, to: i + 1 }
9          ])
10     horizontal[i] = 0;
11 }
12
13
14 for (let i = 0; i < matkul.length; i++) {
15     let shape="";
16     if(matkul[i].wajib){
17         shape="box"
18     }
19     else{
20         shape="circle"
21     }
22
23     horizontal[matkul[i].semester] += 200;
24     nodes.add([
25         id: matkul[i].kode, label: matkul[i].nama, group: matkul[i].semester,
26         x:horizontal[matkul[i].semester], y: matkul[i].semester*150, shape:shape
27     ])
28
29     if(matkul[i].prasyarat.tempuh.length != 0){
30         for (let j = 0; j < matkul[i].prasyarat.tempuh.length; j++){
31             edges.add([
32                 { from: matkul[i].kode, to: matkul[i].prasyarat.tempuh[j],
33                  arrows: "from"
34             })
35         }
36     }
37     if(matkul[i].prasyarat.lulus.length != 0){
38         for (let j = 0; j < matkul[i].prasyarat.lulus.length; j++){
39             edges.add([
40                 { from: matkul[i].kode, to: matkul[i].prasyarat.lulus[j],
41                  dashes: true
42             })
43         }
44     }
45     if(matkul[i].prasyarat.bersamaan.length != 0){
46         for (let j = 0; j < matkul[i].prasyarat.bersamaan.length; j++){
47             edges.add([
48                 { from: matkul[i].kode, to: matkul[i].prasyarat.bersamaan[j],
49                  arrows: "from", dashes: true
50             })
51         }
52     }
53 }
```

- 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 kodennya 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* yang terdapat pada baris satu sampai baris empat. 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* yang terdapat pada baris enam sampai baris empat belas.

Selanjutnya terdapat sebuah *looping* untuk mengisi *array* semester di mana setiap arraynya akan berisi informasi tentang bulan dan tahun untuk setiap semesternya yang terdapat pada baris enam belas sampai baris dua puluh enam.

Kemudian terdapat sebuah *looping* untuk membuat *content* sesuai dengan jumlah mata kuliah, dimana warna, ukuran, dan letak *content* akan diatur di dalamnya yang terdapat pada baris dua puluh delapan sampai baris empat puluh tiga.

Baris empat puluh empat sampai baris lima puluh merupakan kode untuk memasukkan data ke pustaka Vis.js.

Baris lima puluh dua untuk memanggil fungsi yang bernama *fetchUser*.

Untuk kodenya dapat dilihat pada Kode 3.3.

Kode 3.3: Kode untuk membuat Visualisasi Timeline

```

1  var groups = new vis.DataSet([
2      { id: 1, content: "Mata Kuliah Wajib" },
3      { id: 2, content: "Mata Kuliah Pilihan" },
4  ]);
5
6  let tahun=new Date().getFullYear();
7  let bulan = new Date().getMonth();
8  if(bulan<7){
9      tahun-=1
10     bulan=7
11 }
12 else{
13     bulan=7
14 }
15
16 let semester=[];
17 for(i=1;i<=9;i++){
18     semester[i]= tahun.toString() + "-" + bulan.toString() + "-31"
19     if(i%2!=0){
20         bulan+=5
21     }
22     else{
23         bulan-=5
24         tahun+=1
25     }
26 }
27
28 var res = [];
29 let color=["","semester1","semester2","semester3","semester4","semester5","semester6","semester7",
30     ", "semester8"]
31 for (var i = 0; i < matkul.length; i++) {
32     let wajib=,
33     if(matkul[i].wajib){
34         wajib=1;
35     }
36     else{
37         wajib=2;
38     }
39     res.push(
40         { id: matkul[i].kode, content: matkul[i].nama, editable: false, group:wajib ,
41             start: semester[matkul[i].semester], end:semester[matkul[i].semester+1],
42             className: color[matkul[i].semester] }
43     )
44 }
45 var items = new vis.DataSet(res);
46 var container = document.getElementById('timeline');
47 var options = {};
48 var timeline = new vis.Timeline(container, items,groups, options);
49
50 }
51 fetchUsers();
52

```



## BAB 4

# PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan perangkat lunak yang meliputi: perancangan struktur aplikasi dan perancangan antarmuka

### 4.1 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*.
  - **timeline.css**, berkas *css* ini berfungsi untuk memberikan warna pada setiap *content* yang ditampilkan oleh visualisasi *timeline*. Untuk kodenya dapat dilihat pada Kode 4.1

Kode 4.1: Kode *timeline.css*

```
1  .vis-item.semester1 {  
2      background-color: rgb(151, 194, 252);  
3      border-color: rgb(43, 124, 233);  
4      color: black;  
5      box-shadow: 0 0 10px gray;  
6  }  
7  .vis-item.semester2 {  
8      background-color: rgb(255,255,0);  
9      border-color:rgb(255,165,0);  
10     color: black;  
11     box-shadow: 0 0 10px gray;  
12 }  
13 .vis-item.semester3 {  
14     background-color: rgb(251,126,129);  
15     border-color: rgb(251,70,74);  
16     color: black;  
17     box-shadow: 0 0 10px gray;  
18 }  
19 .vis-item.semester4 {  
20     background-color: rgb(123,225,65);  
21     border-color: rgb(76,177,19);  
22     color: black;  
23     box-shadow: 0 0 10px gray;  
24 }  
25 .vis-item.semester5 {  
26     background-color: rgb(235,125,244);  
27     border-color: rgb(225,41,240);  
28     color: black;  
29     box-shadow: 0 0 10px gray;  
30 }  
31 .vis-item.semester6 {  
32     background-color: rgb(173,133,228);  
33     border-color: rgb(124,41,240);  
34     color: black;  
35     box-shadow: 0 0 10px gray;  
36 }  
37 .vis-item.semester7 {  
38     background-color: rgb(255,168,7);  
39     border-color: rgb(217,142,3);  
40     color: black;  
41     box-shadow: 0 0 10px gray;
```

```

42    }
43    .vis-item.semester8 {
44      background-color: #110011;
45      border-color: #663399;
46      color: black;
47      box-shadow: 0 0 10px gray;
48    }

```

- **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 visualisasi dalam bentuk *network*. Fungsi pada berkas ini nantinya akan dipanggil pada *index.html*. Untuk kodennya dapat dilihat pada Kode 4.2.

Kode 4.2: Kode *network.js*

```

1  function processNetwork (){
2    document.getElementById("timeline").style.display="none"
3    document.getElementById("network").style.display="inline"
4    let matkul = '';
5    const fetchUsers = async () => {
6      try {
7        const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json');
8        if (!res.ok) {
9          throw new Error(res.status);
10       }
11       const data = await res.json();
12       matkul = data;
13       createNetwork();
14     } catch (error) {
15       console.log(error);
16     }
17   }
18
19   var nodes = new vis.DataSet();
20
21   var edges = new vis.DataSet();
22
23   function createNetwork() {
24     const horizontal = [];
25     for (i = 1; i < 9; i++) {
26       nodes.add([
27         id: i, label: "Semester" + i, group: i, y: i*150
28       ])
29       if (i != 8) {
30         edges.add([
31           { from: i, to: i + 1 }
32         ])
33       }
34       horizontal[i] = 0;
35     }
36
37     for (let i = 0; i < matkul.length; i++) {
38       let shape="";
39       if(matkul[i].wajib){
40         shape="box"
41       }
42       else{
43         shape="circle"
44       }
45
46       horizontal[matkul[i].semester] += 200;
47       nodes.add([
48         id: matkul[i].kode, label: matkul[i].nama, group: matkul[i].semester,
49         x:horizontal[matkul[i].semester], y: matkul[i].semester*150, shape:shape
50       ])
51
52       if(matkul[i].prasyarat.tempuh.length != 0){
53         for (let j = 0; j < matkul[i].prasyarat.tempuh.length; j++){
54           edges.add([
55             { from: matkul[i].kode, to: matkul[i].prasyarat.tempuh[j],
56               arrows: "from"
57             })
58         }
59       }
59       if(matkul[i].prasyarat.lulus.length != 0){
60         for (let j = 0; j < matkul[i].prasyarat.lulus.length; j++){
61           edges.add([
62             { from: matkul[i].kode, to: matkul[i].prasyarat.lulus[j],
63               dashes: true
64             })
65         }
66       }
67       if(matkul[i].prasyarat.bersamaan.length != 0){
68         for (let j = 0; j < matkul[i].prasyarat.bersamaan.length; j++){
69           edges.add([
70             { from: matkul[i].kode, to: matkul[i].prasyarat.bersamaan[j],
71               arrows: "from", dashes: true
72             })
73         }
73       }
74     }
75   }

```

```

74        }
75    }
76  }
77}
78fetchUsers();
79
80var container = document.getElementById('network');
81
82var data = {
83  nodes: nodes,
84  edges: edges
85};
86
87var options = {
88  nodes: {
89    margin: 10,
90    widthConstraint: {
91      maximum: 120,
92    },
93  },
94  physics: false,
95  edges: {
96    smooth: false
97  },
98};
99
100var network = new vis.Network(container, data, options);
101

```

Untuk membedakan mata kuliah berdasarkan semesternya akan diberikan warna yang berbeda pada setiap *nodenya*, yaitu:

- \* 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.

Kemudian, terdapat tiga macam edges yang warnanya mengikuti warna *nodenya*, yaitu :

1. edges yang berbentuk garis dengan ujung anak panah yang melambangkan prasyarat tempuh.
  2. edges yang berbentuk garis putus - putus saja yang melambangkan prasyarat lulus.
  3. edges yang berbentuk garis putus - putus dengan ujung anak panah yang melambangkan prasyarat bersamaan.
- **timeline.js**, berkas *JavaScript* ini berisi fungsi khusus untuk membuat visualisasi dalam bentuk *timeline*. Fungsi pada berkas ini nantinya akan dipanggil pada index.html. Untuk kodennya dapat dilihat pada Kode 4.3.

Kode 4.3: Kode *timeline.js*

```

1  function processTimeline(){
2    document.getElementById("network").style.display="none"
3    document.getElementById("timeline").style.display="inline"
4    let matkul = '',
5    const fetchUsers = async () => {
6      try {
7        const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json');
8        if (!res.ok) {
9          throw new Error(res.status);
10       }
11       const data = await res.json();
12       matkul = data;
13       createTimeline(matkul);
14     } catch (error) {
15       console.log(error);
16     }
17   }
18
19   function createTimeline() {
20     var groups = new vis.DataSet([
21       { id: 1, content: "Mata Kuliah Wajib" },
22       { id: 2, content: "Mata Kuliah Pilihan" },
23     ]);
24
25     let tahun=new Date().getFullYear();
26     let bulan = new Date().getMonth();
27     if(bulan<7){
28       tahun-=1
29       bulan=7
30     }

```

```

31         else{
32             bulan=7
33         }
34
35     let semester=[];
36     for(i=1;i<=9;i++){
37         semester[i]= tahun.toString() + "-" + bulan.toString() + "-31"
38         if(i%2!=0){
39             bulan+=5
40         }
41         else{
42             bulan-=5
43             tahun+=1
44         }
45     }
46
47     var res = [];
48     let let color=["","semester1","semester2","semester3","semester4","semester5","semester6","semester7",
49     "semester8"]
50     for (var i = 0; i < matkul.length; i++) {
51         let wajib="";
52         if(matkul[i].wajib){
53             wajib=1;
54         }
55         else{
56             wajib=2;
57         }
58         res.push(
59             { id: matkul[i].kode, content: matkul[i].nama, editable: false, group:wajib ,
60             start: semester[matkul[i].semester],end:semester[matkul[i].semester+1],
61             className: color[matkul[i].semester] }
62         )
63     var items = new vis.DataSet(res);
64
65     var container = document.getElementById('timeline');
66
67     var options = {};
68
69     var timeline = new vis.Timeline(container, items,groups, options);
70
71 }
72 fetchUsers();
    
```

Sama seperti *network*, untuk membedakan mata kuliah berdasarkan semesternya akan diberikan warna kepada setiap *contentnya*, yaitu:

- \* 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.
- **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*

```

1 <html>
2   <head>
3     <script type="text/javascript" src="js/vis.js"></script>
4     <link href="css/vis-network.min.css" rel="stylesheet" type="text/css" />
5     <link href="css/vis-timeline-graph2d.min.css" rel="stylesheet" type="text/css" />
6
7     <script type="text/javascript" src="js/network.js"></script>
8     <script type="text/javascript" src="js/timeline.js"></script>
9   </head>
10
11  <body><br>
12    <button type="button" onclick="processNetwork()">Network</button>
13    <button type="button" onclick="processTimeline()">Timeline</button><br><br>
14    <div id="network"></div>
15    <div id="timeline"></div>
16
17  </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*

```

1 const { app, BrowserWindow } = require('electron')
2 const path = require('path')
3
4 function createWindow () {
5   const win = new BrowserWindow({
6     width: 800,
7     height: 600,
8     webPreferences: {
9       preload: path.join(__dirname, 'preload.js')
10    }
11  })
12
13  win.loadFile('index.html')
14
15 app.whenReady().then(() => {
16   createWindow()
17 })
18
19 app.on('window-all-closed', () => {
20   if (process.platform !== 'darwin') {
21     app.quit()
22   }
23 })
24

```

- **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*

```

1 {
2   "name": "VisKur",
3   "version": "1.0.0",
4   "description": "Visualisasi_Kurikulum",
5   "main": "main.js",
6   "scripts": {
7     "start": "electron ."
8   },
9   "author": "Joshua_Delavo",
10  "license": "MIT",
11  "devDependencies": {
12    "electron": "^16.0.0"
13  }
14

```

- **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*

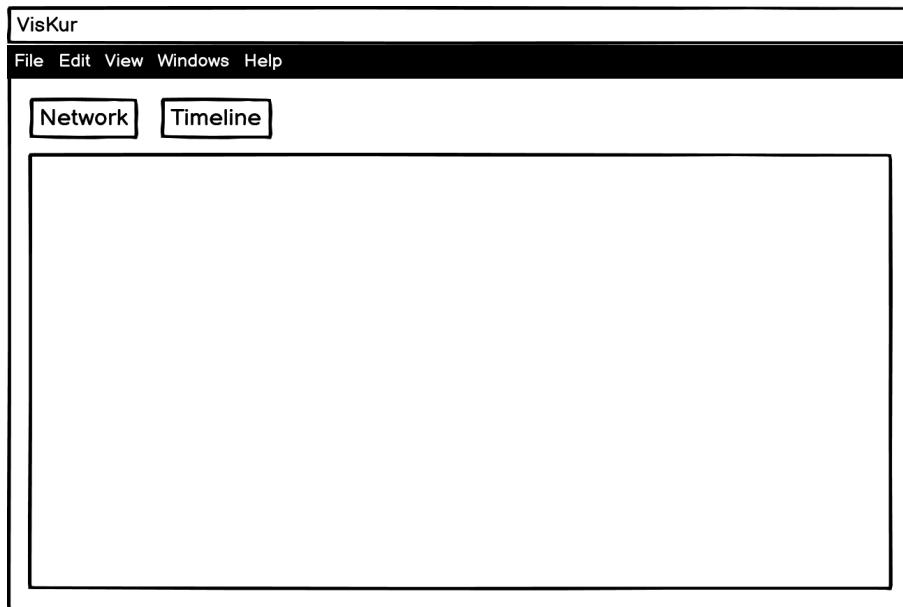
```

1 window.addEventListener('DOMContentLoaded', () => {
2   const replaceText = (selector, text) => {
3     const element = document.getElementById(selector)
4     if (element) element.innerText = text
5   }
6
7   for (const type of ['chrome', 'node', 'electron']) {
8     replaceText(`${type}-version', process.versions[type])
9   }
10

```

## 4.2 Perancangan Antarmuka

Perancangan antarmuka dibuat sesuai kebutuhan pengguna Aplikasi VisKur. Antarmuka ini hanya akan mengeluarkan antarmuka keluaran seperti pada gambar 4.1. Berikut ini merupakan rancangan antarmuka aplikasi yang akan dibuat:



Gambar 4.1: Rancangan Antarmuka Aplikasi VisKur

Rancangan antarmuka ini merupakan 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 visualisasi 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.

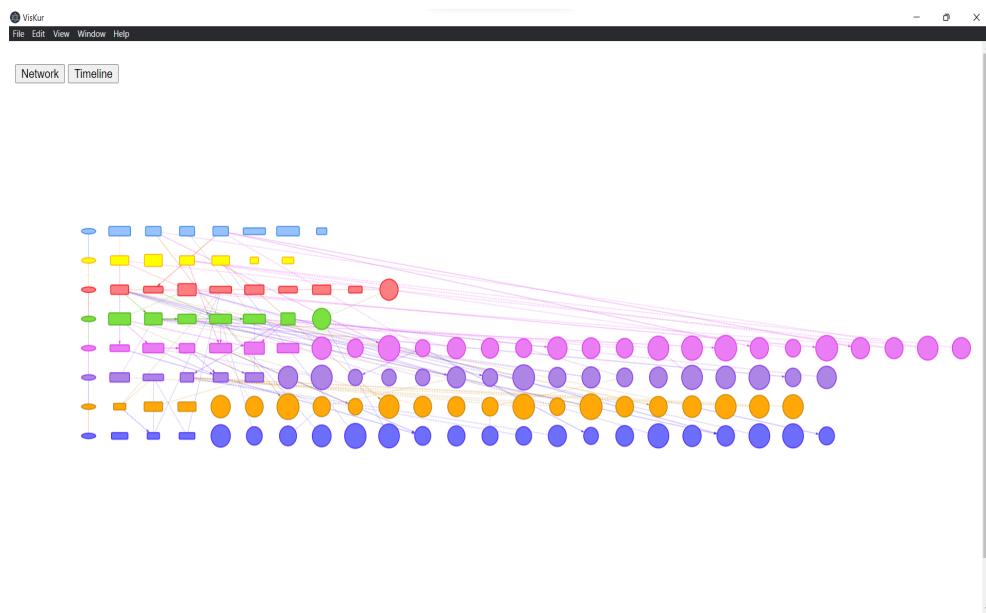
## BAB 5

# IMPLEMENTASI DAN PENGUJIAN

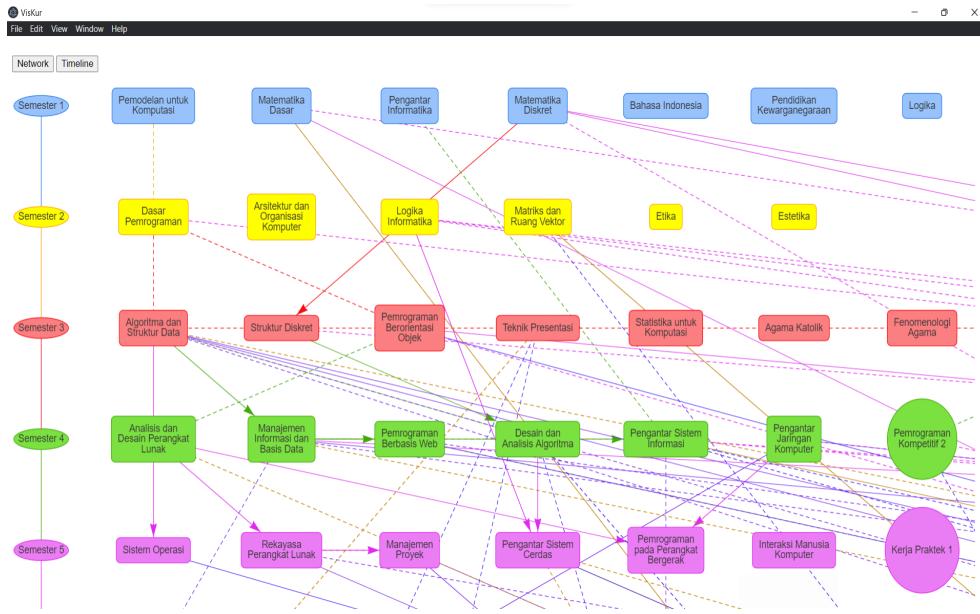
### 5.1 Implementasi

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

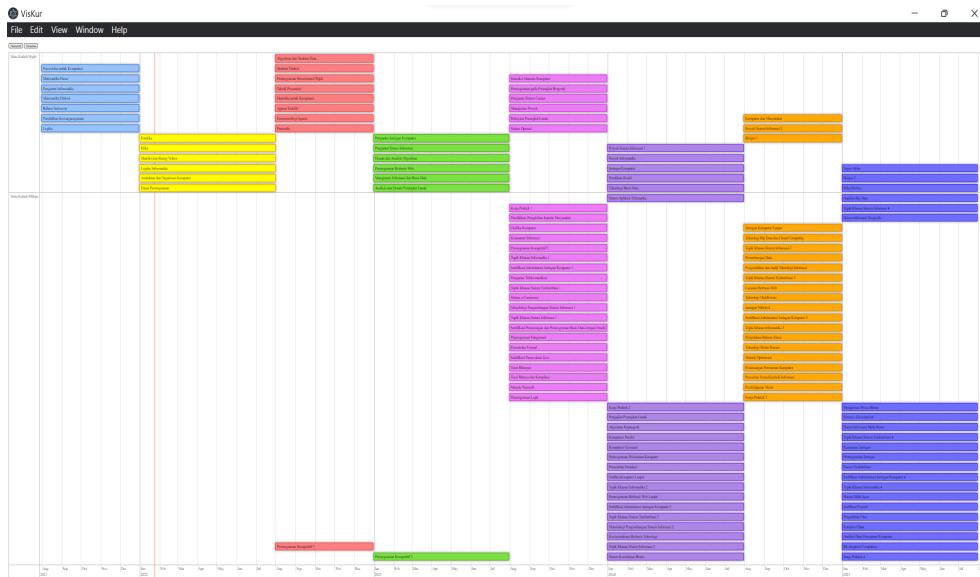
- Hasil visualisasi *Network* secara keseluruhan dapat dilihat pada Gambar 5.1, kemudian untuk lebih jelasnya dapat dilihat pada Gambar 5.2.

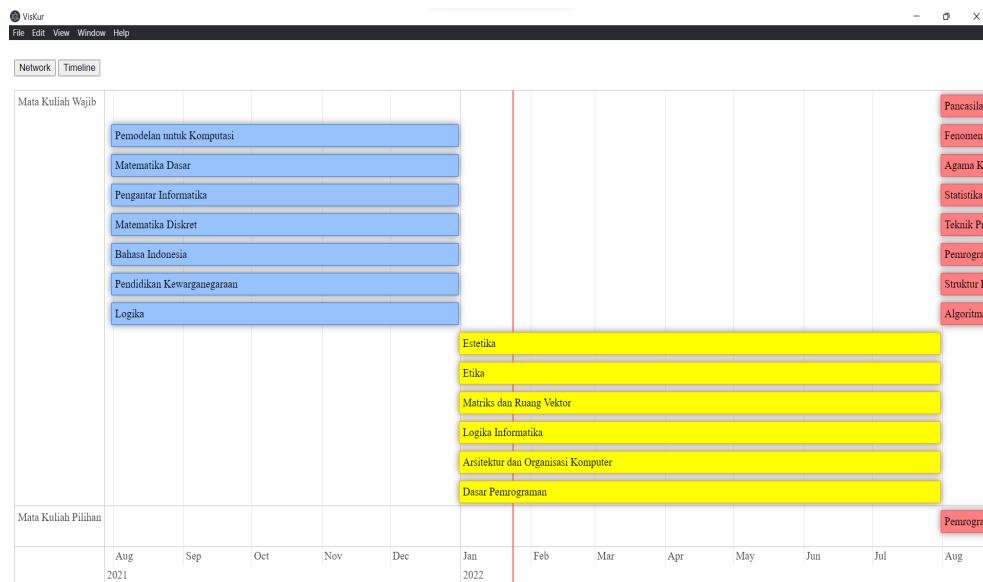


Gambar 5.1: Hasil visualisasi *Network* secara keseluruhan

Gambar 5.2: Hasil visualisasi *Network* secara lebih jelas

- Hasil visualisasi *Timeline* secara keseluruhan dapat dilihat pada Gambar 5.3, kemudian untuk lebih jelasnya dapat dilihat pada Gambar 5.4.

Gambar 5.3: Hasil visualisasi *Timeline* secara keseluruhan



Gambar 5.4: Hasil visualisasi *Timeline* secara lebih jelas

## 5.2 Pengujian

### 5.2.1 Lingkungan Pengujian

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

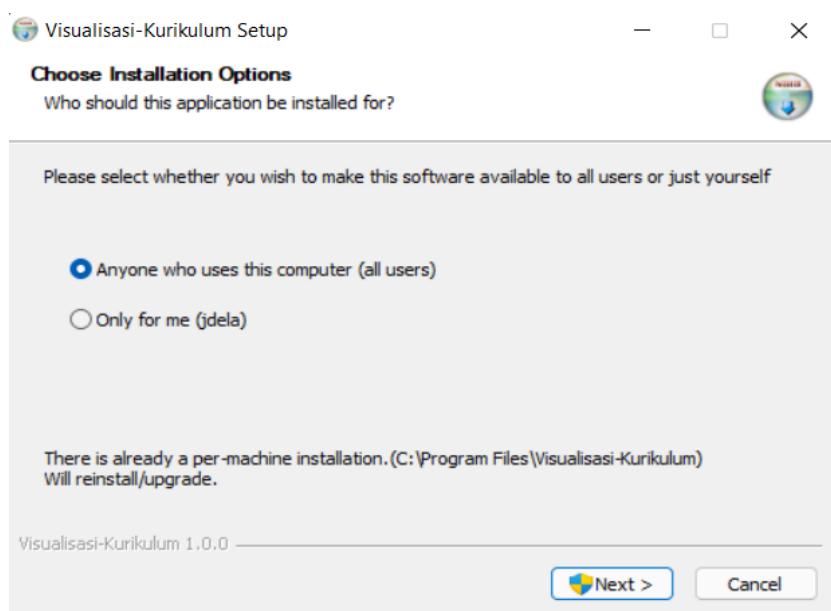
1. Sistem Operasi: Windows 11
2. Versi Node: 14.17.6
3. Versi NPM: 6.14.15
4. Versi *Electron*: 16.0.0

### 5.2.2 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui keberhasilan pemasangan perangkat lunak di komputer penulis.

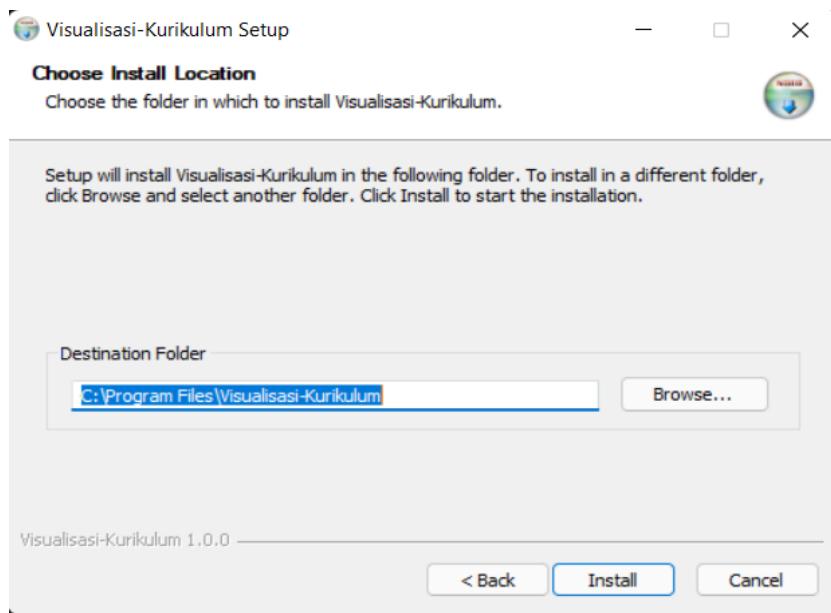
Berikut merupakan langkah - langkah yang dilakukan untuk memasang serta menggunakan aplikasi VisKur:

- 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.



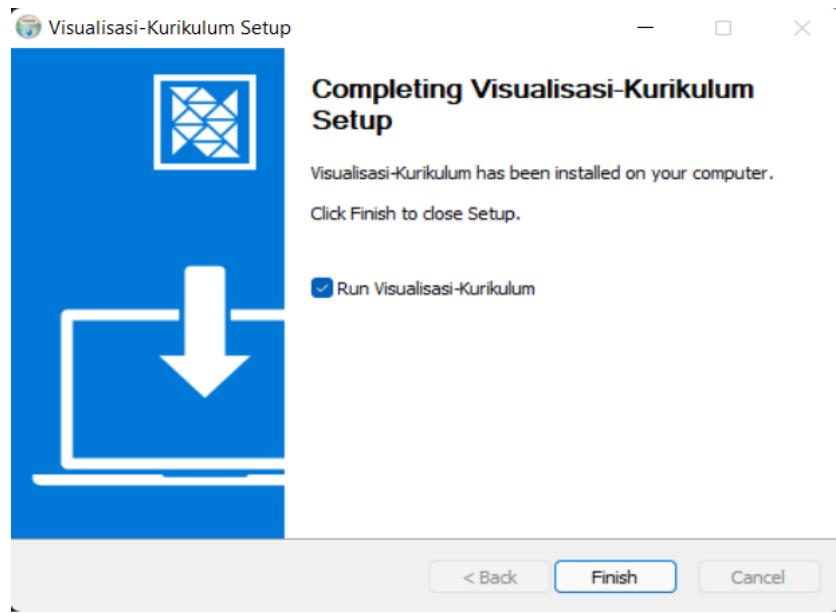
Gambar 5.5: *Setup* aplikasi VisKur

- 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.



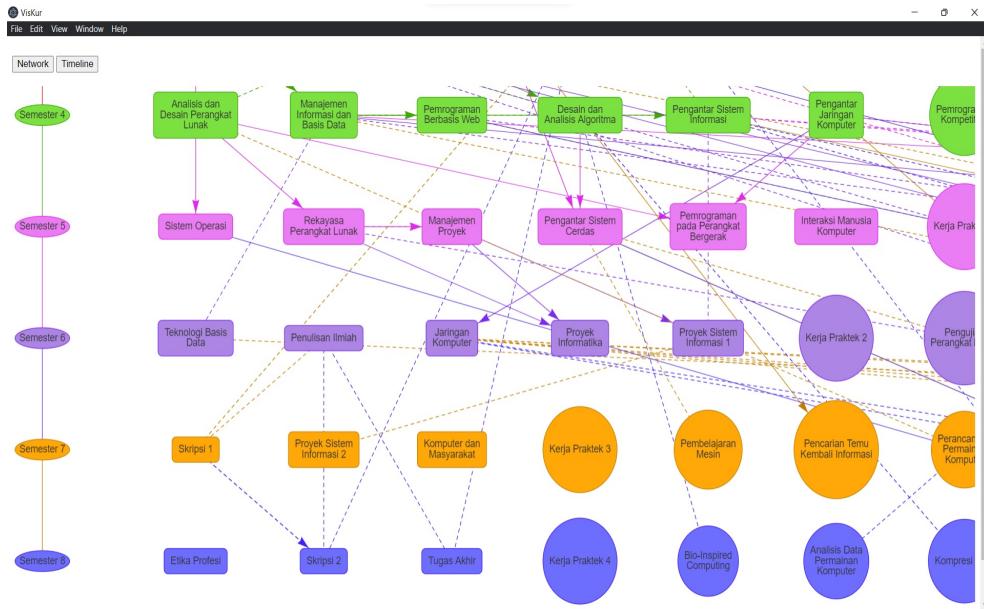
Gambar 5.7: *Setup* aplikasi VisKur

- Setelah aplikasi dijalankan, maka aplikasi akan terlihat seperti pada Gambar 5.8, kemudian saat menekan tombol *Network* akan terlihat seperti pada Gambar 5.2, lalu jika menekan tombol *Timeline* akan terlihat seperti pada Gambar 5.4.

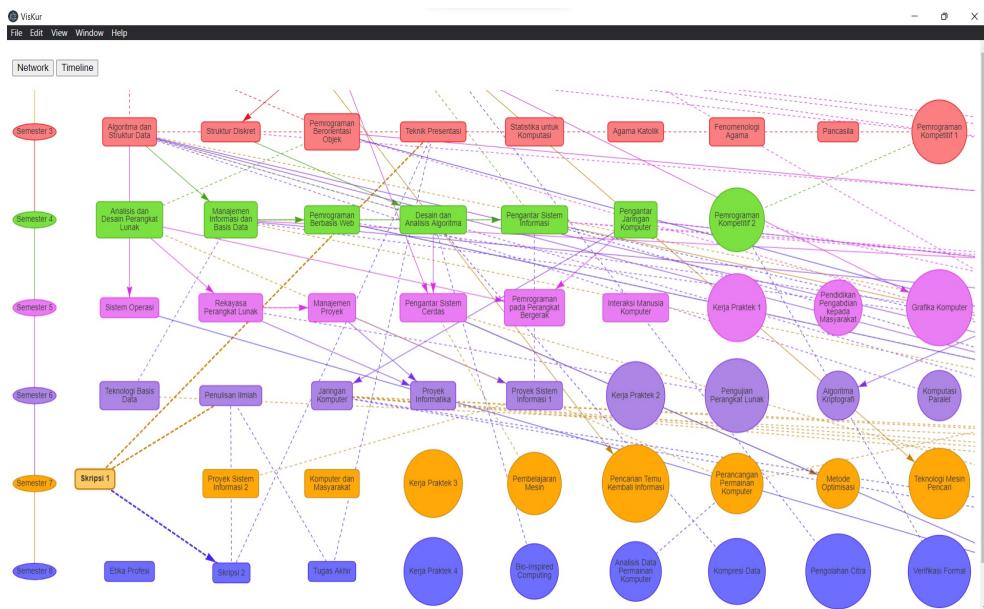


Gambar 5.8: Halaman utama aplikasi VisKur.

Pada visualisasi *network*, *node* dapat ditekan dan sambil menekannya dapat digeser ke semua arah sesuai dengan kemauan pengguna. Berikut merupakan contoh penggeseran *node* skripsi 1 ke kiri yang sebelum digeser dapat dilihat pada Gambar 5.9 dan setelah digeser dapat dilihat pada Gambar 5.10.

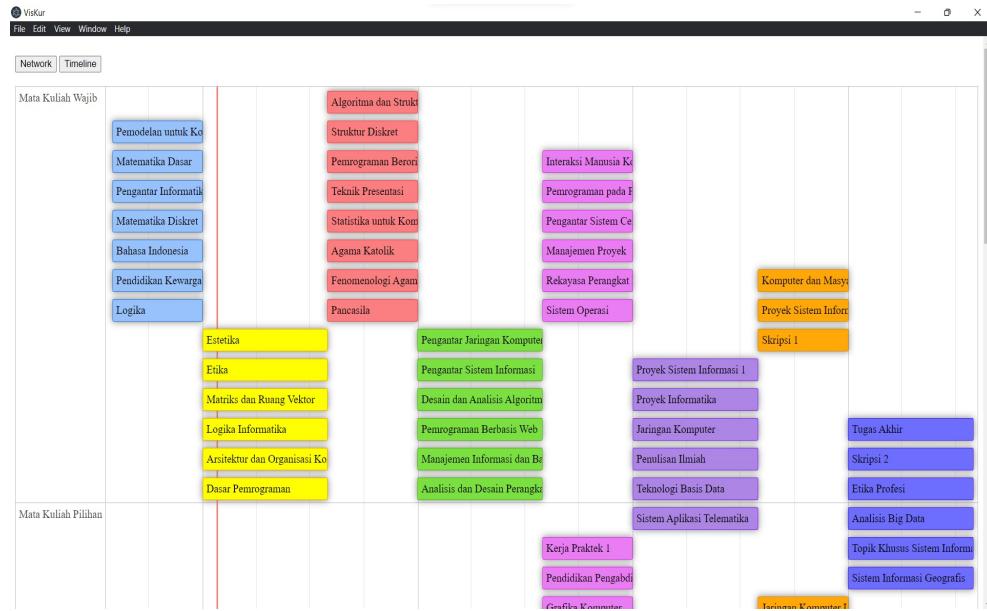
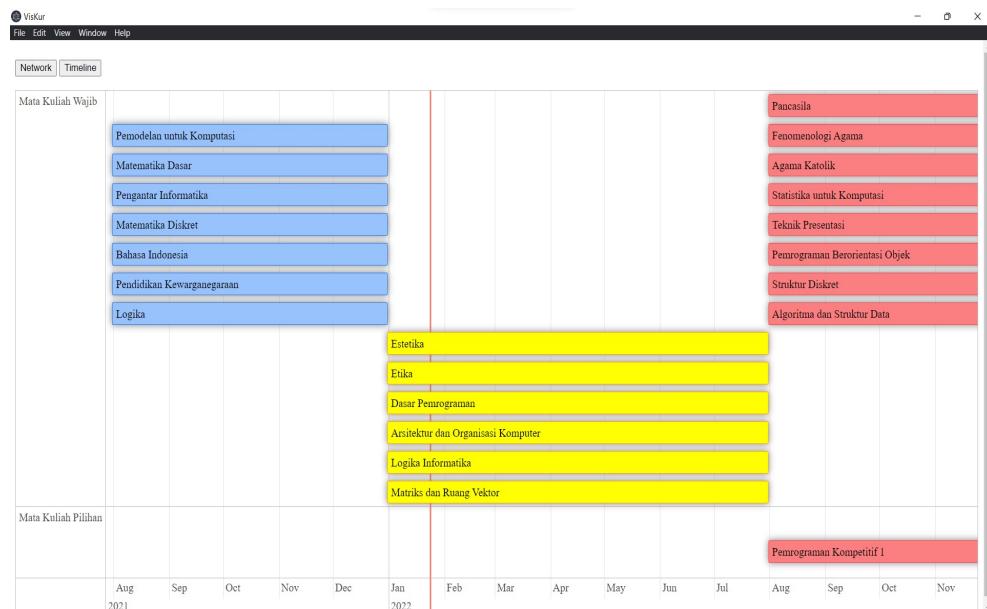


Gambar 5.9: Node skripsi 1 sebelum digeser



Gambar 5.10: Node skripsi 1 setelah ditekan dan digeser ke kiri

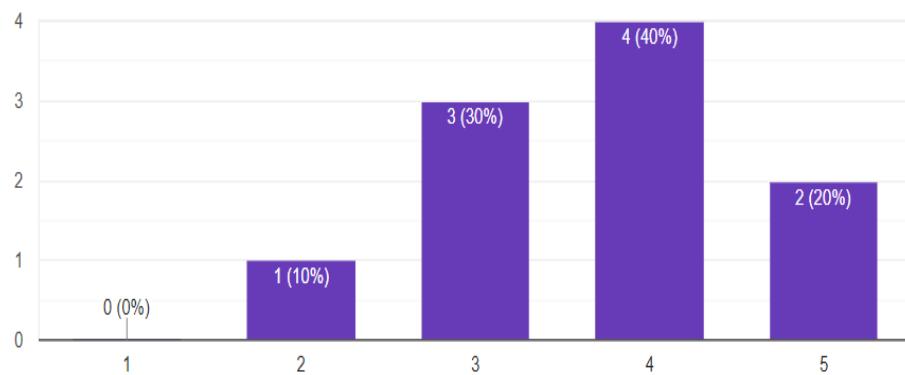
Pada visualisasi *timeline*, pengguna dapat melakukan gulir ke atas untuk memperbesar tampilan dan gulir ke bawah untuk memperkecil tampilan, kemudian pengguna juga dapat melakukan penggeseran *timeline* untuk melihat mata kuliah di semester selanjutnya. Berikut merupakan contoh gulir atas (memperbesar tampilan) dan penggeseran *timeline* yang sebelumnya dapat dilihat pada Gambar 5.11 dan setelahnya dapat dilihat pada Gambar 5.12.

Gambar 5.11: *Timeline* sebelum digulir dan digeserGambar 5.12: *Timeline* setelah digulir dan digeser

### 5.2.3 Pengujian Ekperimental

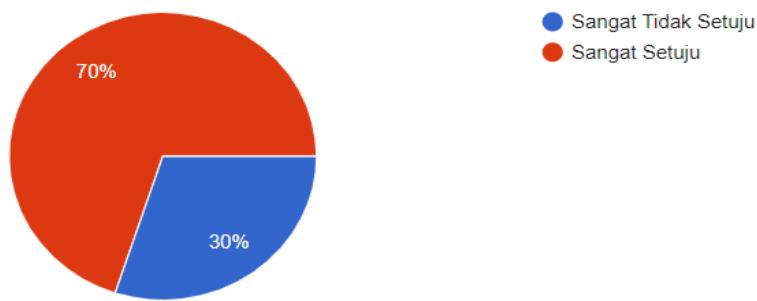
Pengujian eksperimental pada skripsi ini akan dilakukan dengan melakukan survei kepada sepuluh orang mahasiswa aktif Universitas Katolik Parahyangan jurusan informatika yang kemudian didapatkan hasil seperti berikut:

- Hasil diagram batang pada Gambar 5.13 menunjukkan bahwa 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.



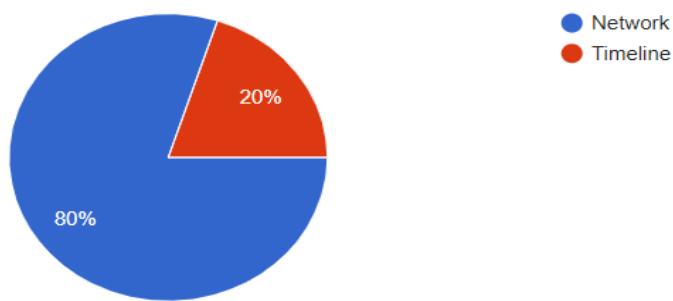
Gambar 5.13: Diagram batang hasil pemilihan aplikasi VisKur terhadap pohon kurikulum

- Hasil diagram lingkaran pada Gambar 5.14 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 5.14: 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 membungkung 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 selengkap informasi yang ditampilkan pada pohon kurikulum karena tidak ada jumlah sks untuk setiap mata kuliahnya.
- Hasil dari diagram lingkaran pada Gambar 5.15 menunjukkan bahwa delapan puluh persen mahasiswa memilih jenis visualisasi *Network* dan dua puluh persen mahasiswa memilih jenis visualisasi *Timeline* yang lebih cocok untuk memvisualisasikan kurikulum 2018.



Gambar 5.15: Diagram lingkaran pemilihan bentuk *Network* dengan bentuk *Timeline*

- 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 ke beberapa mahasiswa Teknik Informatika jurusan 2017 bahwa sebagian besar dari mereka telah setuju dengan aplikasi VisKur yang dapat membantu mereka untuk melihat kurikulum dengan lebih jelas ketimbang melihat pohon kurikulum yang terdapat pada buku petunjuk pelaksanaan kegiatan akademik (juklak).
- Aplikasi VisKur telah dapat mengambil data dari *API* kemudian membuatkan visualisasinya dalam bentuk *Network* dan *Timeline*.
- Aplikasi VisKur telah dapat dipasang dan dijalankan pada komputer dengan 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 agar lebih menarik, sesuai dengan masukan dari survei ke beberapa pengguna.
- Penambahan fitur saran pengambilan matakuliah untuk mahasiswa 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] Vis.js. <https://visjs.org/>. Accessed: 2021-04-27.
- [4] Ftis open data. <https://github.com/ftisunpar/data>. Accessed: 2021-04-27.
- [5] Javascript. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Accessed: 2021-10-29.
- [6] Async wait. [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async\\_await](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await). Accessed: 2021-10-29.



# LAMPIRAN A

## KODE PROGRAM

Kode A.1: timeline.css

```
1 .vis-item.semester1 {
2     background-color: rgb(151, 194, 252);
3     border-color: rgb(43, 124, 233);
4     color: black;
5     box-shadow: 0 0 10px gray;
6 }
7 .vis-item.semester2 {
8     background-color: rgb(255,255,0);
9     border-color:rgb(255,165,0);
10    color: black;
11    box-shadow: 0 0 10px gray;
12 }
13 .vis-item.semester3 {
14     background-color: rgb(251,126,129);
15     border-color: rgb(251,70,74);
16     color: black;
17     box-shadow: 0 0 10px gray;
18 }
19 .vis-item.semester4 {
20     background-color: rgb(123,225,65);
21     border-color: rgb(76,177,19);
22     color: black;
23     box-shadow: 0 0 10px gray;
24 }
25 .vis-item.semester5 {
26     background-color: rgb(235,125,244);
27     border-color: rgb(225,41,240);
28     color: black;
29     box-shadow: 0 0 10px gray;
30 }
31 .vis-item.semester6 {
32     background-color: rgb(173,133,228);
33     border-color: rgb(124,41,240);
34     color: black;
35     box-shadow: 0 0 10px gray;
36 }
37 .vis-item.semester7 {
38     background-color: rgb(255,168,7);
39     border-color: rgb(217,142,3);
40     color: black;
41     box-shadow: 0 0 10px gray;
42 }
43 .vis-item.semester8 {
44     background-color: rgb(110,110,253);
45     border-color: rgb(68,35,251);
46     color: black;
47     box-shadow: 0 0 10px gray;
48 }
```

Kode A.2: network.js

```
1 function processNetwork (){
2     document.getElementById("timeline").style.display="none"
3     document.getElementById("network").style.display="inline"
4     let matkul = '';
5     const fetchUsers = async () => {
6         try {
7             const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json');
8             if (!res.ok) {
9                 throw new Error(res.status);
10            }
11            const data = await res.json();
12            matkul = data;
13            createNetwork();
14        } catch (error) {
15            console.log(error);
16        }
17    }
18    var nodes = new vis.DataSet();
19    var edges = new vis.DataSet();
20
21}
```

```

23|     function createNetwork() {
24|         const horizontal = [];
25|         for (i = 1; i < 9; i++) {
26|             nodes.add([
27|                 id: i, label: "Semester" + i, group: i, y: i*150
28|             ])
29|             if (i != 8) {
30|                 edges.add([
31|                     { from: i, to: i + 1 }
32|                 ])
33|             }
34|             horizontal[i] = 0;
35|         }
36|
37|         for (let i = 0; i < matkul.length; i++) {
38|             let shape="";
39|             if(matkul[i].wajib){
40|                 shape="box"
41|             }
42|             else{
43|                 shape="circle"
44|             }
45|
46|             horizontal[matkul[i].semester] += 200;
47|             nodes.add([
48|                 id: matkul[i].kode, label: matkul[i].nama, group: matkul[i].semester,
49|                 x:horizontal[matkul[i].semester], y: matkul[i].semester*150, shape:shape
50|             ])
51|
52|             if(matkul[i].prasyarat.tempuh.length != 0){
53|                 for (let j = 0; j < matkul[i].prasyarat.tempuh.length; j++){
54|                     edges.add([
55|                         { from: matkul[i].kode, to: matkul[i].prasyarat.tempuh[j],
56|                           arrows: "from"
57|                         })
58|                 }
59|             }
60|             if(matkul[i].prasyarat.lulus.length != 0){
61|                 for (let j = 0; j < matkul[i].prasyarat.lulus.length; j++){
62|                     edges.add([
63|                         { from: matkul[i].kode, to: matkul[i].prasyarat.lulus[j],
64|                           dashes: true
65|                         })
66|                 }
67|             }
68|             if(matkul[i].prasyarat.bersamaan.length != 0){
69|                 for (let j = 0; j < matkul[i].prasyarat.bersamaan.length; j++){
70|                     edges.add([
71|                         { from: matkul[i].kode, to: matkul[i].prasyarat.bersamaan[j],
72|                           arrows: "from", dashes: true
73|                         })
74|                 }
75|             }
76|         }
77|     }
78|     fetchUsers();
79|
80|     var container = document.getElementById('network');
81|
82|     var data = {
83|         nodes: nodes,
84|         edges: edges
85|     };
86|
87|     var options = {
88|         nodes: {
89|             margin: 10,
90|             widthConstraint: {
91|                 maximum: 120,
92|             },
93|         },
94|         physics: false,
95|         edges: {
96|             smooth: false
97|         },
98|     };
99|
100|     var network = new vis.Network(container, data, options);
101}

```

Kode A.3: timeline.js

```

1 function processTimeline(){
2     document.getElementById("network").style.display="none"
3     document.getElementById("timeline").style.display="inline"
4     let matkul = ''
5     const fetchUsers = async () => {
6         try {
7             const res = await fetch('https://raw.githubusercontent.com/ftisunpar/data/master/prasyarat.json');
8             if (!res.ok) {
9                 throw new Error(res.status);
10            }
11            const data = await res.json();
12            matkul = data;
13            createTimeline(matkul);
14        } catch (error) {
15            console.log(error);
16        }
17    }

```

```

17| }
18|
19| function createTimeline() {
20|     var groups = new vis.DataSet([
21|         { id: 1, content: "Mata Kuliah Wajib" },
22|         { id: 2, content: "Mata Kuliah Pilihan" },
23|     ]);
24|
25|     let tahun=new Date().getFullYear();
26|     let bulan = new Date().getMonth();
27|     if(bulan<7){
28|         tahun-=1
29|         bulan=7
30|     }
31|     else{
32|         bulan=7
33|     }
34|
35|     let semester=[];
36|     for(i=1;i<=9;i++){
37|         semester[i]= tahun.toString()+"-"+bulan.toString()+"-31"
38|         if(i%2!=0){
39|             bulan+=5
40|         }
41|         else{
42|             bulan-=5
43|             tahun+=1
44|         }
45|     }
46|
47|     var res = [];
48|     let color=["","semester1","semester2","semester3","semester4","semester5","semester6","semester7","semester8"]
49|     for (var i = 0; i < matkul.length; i++) {
50|         let wajib="";
51|         if(matkul[i].wajib){
52|             wajib=1;
53|         }
54|         else{
55|             wajib=2;
56|         }
57|         res.push(
58|             { id: matkul[i].kode, content: matkul[i].nama, editable: false, group:wajib ,
59|               start: semester[matkul[i].semester], end:semester[matkul[i].semester+1],
60|               className: color[matkul[i].semester] }
61|         )
62|     }
63|     var items = new vis.DataSet(res);
64|
65|     var container = document.getElementById('timeline');
66|
67|     var options = {};
68|
69|     var timeline = new vis.Timeline(container, items,groups, options);
70|
71|     fetchUsers();
72}

```

Kode A.4: index.html

```

1<html>
2<head>
3    <script type="text/javascript" src="js/vis.js"></script>
4    <link href="css/vis-network.min.css" rel="stylesheet" type="text/css" />
5    <link href="css/vis-timeline-graph2d.min.css" rel="stylesheet" type="text/css" />
6    <link href="css/timeline.css" rel="stylesheet" type="text/css" />
7
8    <script type="text/javascript" src="js/network.js"></script>
9    <script type="text/javascript" src="js/timeline.js"></script>
10</head>
11
12<body><br>
13    <button type="button" onclick="processNetwork()">Network</button>
14    <button type="button" onclick="processTimeline()">Timeline</button><br><br>
15    <div id="network"></div>
16    <div id="timeline"></div>
17</body>
18</html>

```

Kode A.5: main.js

```

1 const { app, BrowserWindow } = require('electron')
2 const path = require('path')
3
4 function createWindow () {
5     const win = new BrowserWindow({
6         width: 800,
7         height: 600,
8         webPreferences: {
9             preload: path.join(__dirname, 'preload.js')
10        }
11    })
12
13    win.loadFile('index.html')
14 }
15

```

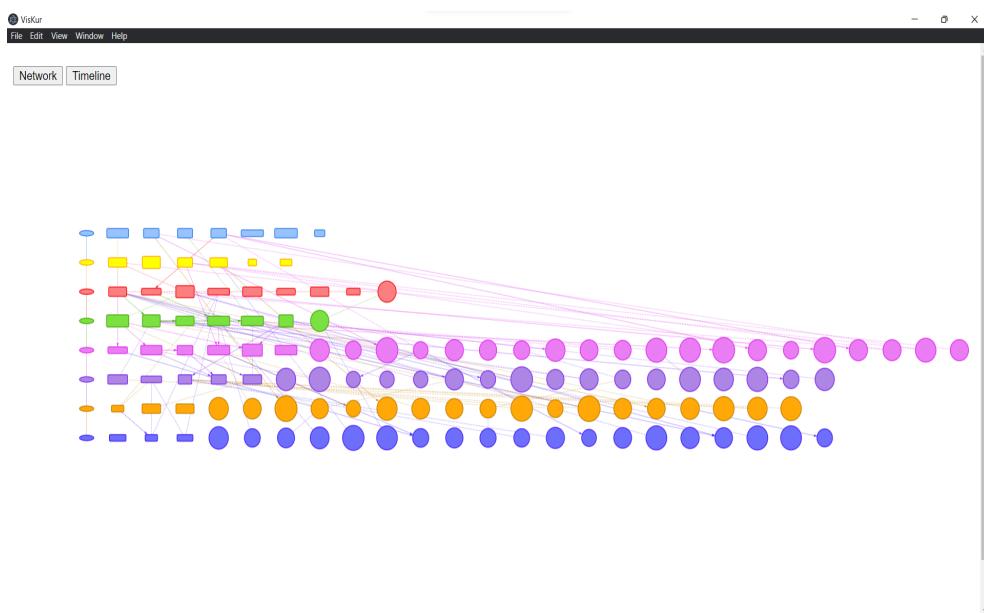
```
16| app.whenReady().then(() => {
17|   createWindow()
18| }
19|
20| app.on('window-all-closed', () => {
21|   if (process.platform !== 'darwin') {
22|     app.quit()
23|   }
24| })
```

Kode A.6: preload.js

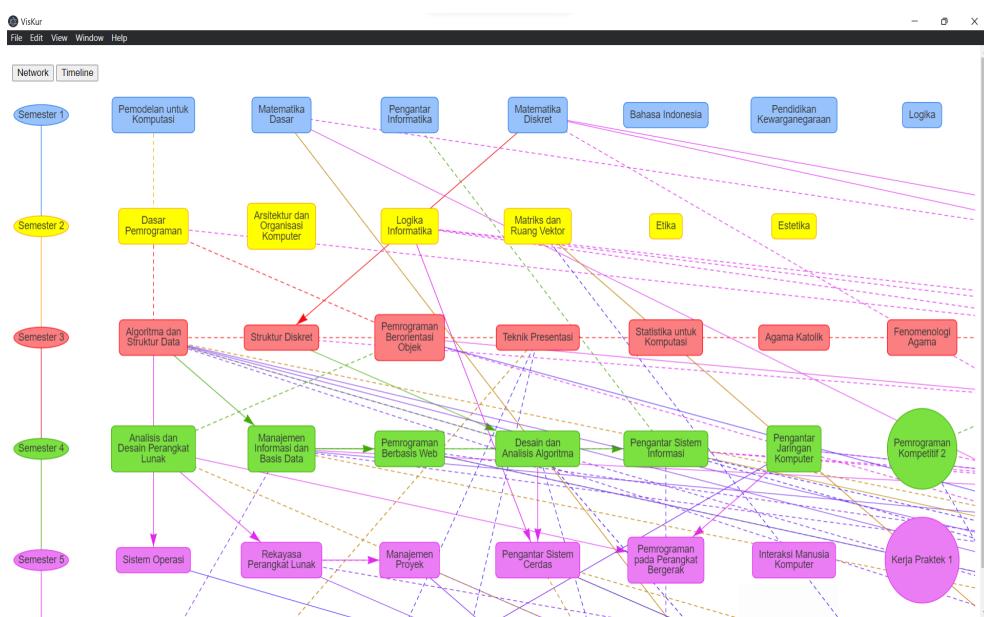
```
1 window.addEventListener('DOMContentLoaded', () => {
2   const replaceText = (selector, text) => {
3     const element = document.getElementById(selector)
4     if (element) element.innerText = text
5   }
6
7   for (const type of ['chrome', 'node', 'electron']) {
8     replaceText(`#${type}-version`, process.versions[type])
9   }
10 })
```

## LAMPIRAN B

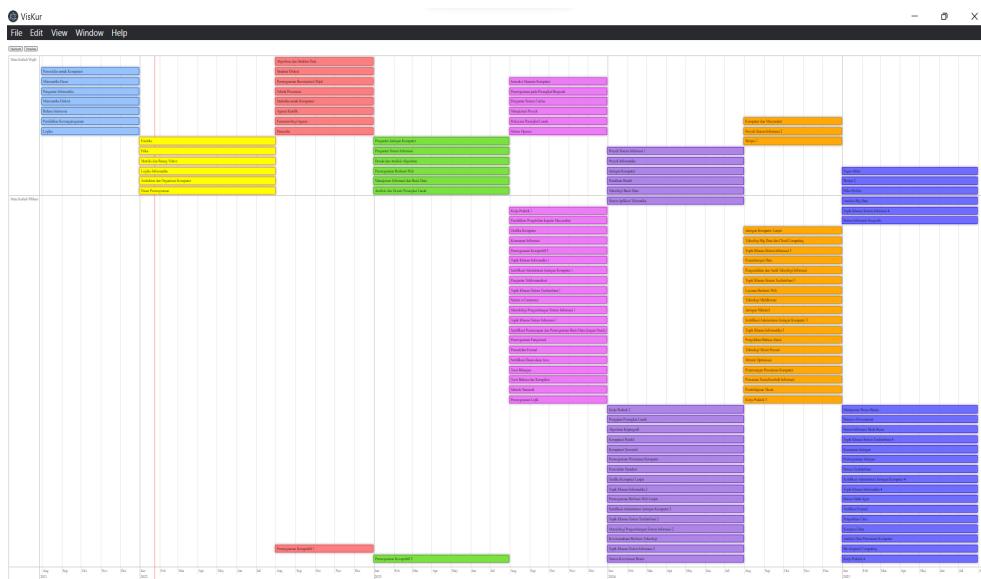
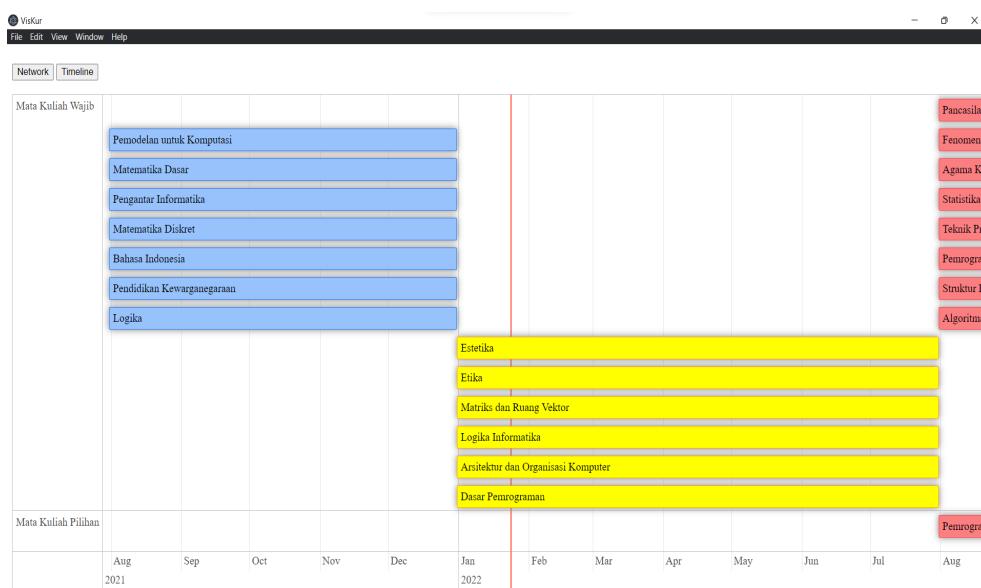
### HASIL EKSPERIMENT



Gambar B.1: Hasil visualisasi *Network* secara keseluruhan



Gambar B.2: Hasil visualisasi *Network* secara lebih jelas

Gambar B.3: Hasil visualisasi *Timeline* secara keseluruhanGambar B.4: Hasil visualisasi *Timeline* secara lebih jelas