Laporan

Tugas Kecil 3 IF2211 Strategi Algoritma

Implementasi Algoritma A* untuk Menentukan Lintasan Terpendek

oleh

Andres Jerriel Sinabutar - 13519218

Ryo Richardo - 13519193

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

**1. *Source Code* dalam Bahasa Pemrograman Javascript **

```
// Algoritma A*
// Inisialisasi elemen pertama dan queue
export default function aStar(idxsrc, idxdes, matrix, coor) {
    // Kamus
    // idxsrc   : indeks start node
    // idxdes   : indeks destination node
    // matrix   : adjacency matrix
    // long     : list of longitudes
    // lat      : list of lattitudes


    let el = new Element(idxsrc, [], 0, 0);
    let queue = new PriorityQueue();
    queue.enqueue(el);


    // Pengulangan sampai top of queue adalah node tujuan atau
queue kosong
    while (el.getIdx() !== idxdes && !queue.isEmpty()) {
        el = queue.dequeue();
        el.addNode(el.getIdx());


        // Mencari adjacency di matrix
        for (let kolom = 0; kolom < matrix[el.getIdx()].length;
kolom++) {
            if (matrix[el.getIdx()][kolom] === 1
&& !queue.hasVisited.includes(kolom)) {
                let distanceElNextEl =
haversineDistance(coor[el.getIdx()][0], coor[el.getIdx()][1],
coor[kolom][0], coor[kolom][1]); // hitung jarak pindah elemen
                let heuristicDistance =
haversineDistance(coor[idxdes][0], coor[idxdes][1],
coor[kolom][0], coor[kolom][1]); // hitung jarak heuristik nextEl
ke tujuan
```

```
                let newWeightVisited = el.getWeightVisited() +
distanceElNextEl; // jarak baru (g(n))
                let newWeight = newWeightVisited +
heuristicDistance; // bobot baru (g(n) + h(n))
                queue.enqueue(new Element(kolom, el.getVisited(),
newWeightVisited, newWeight)); // enqueue
            }
        }
        // Next element
        if (!queue.isEmpty()) {
            el = queue.peek();
        }
    }

    // Isi el.visited dengan diri sendiri
    el.addNode(el.getIdx());

    // Kondisi ketemu
    if (el.getIdx() === idxdes) {
        let res = [];
        el.visited.forEach(element => {
            res.push(coor[element]);
        })
        return res;
    }

    // Kondisi gaketemu array kosong
    else {
        return [];
    }
}
```

2. Peta/Graf Input

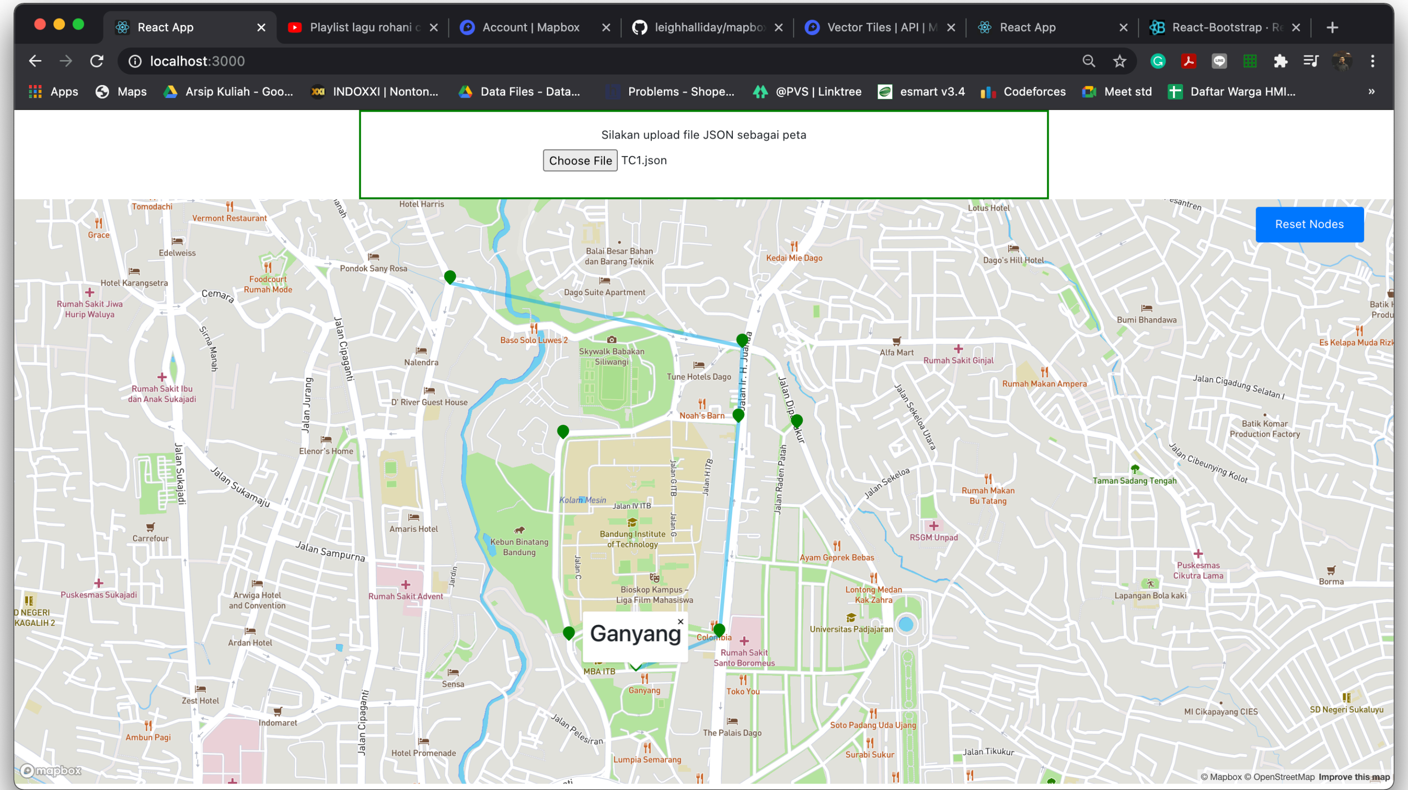
Peta/graf input di*input* dalam format JSON yang berisi nodes letak koordinat tiap tempat yang akan dijadikan graf dan adjacency matrix sebagai representasi keterhubungan antar simpul. Contoh input peta yang dimasukkan seperti berikut:

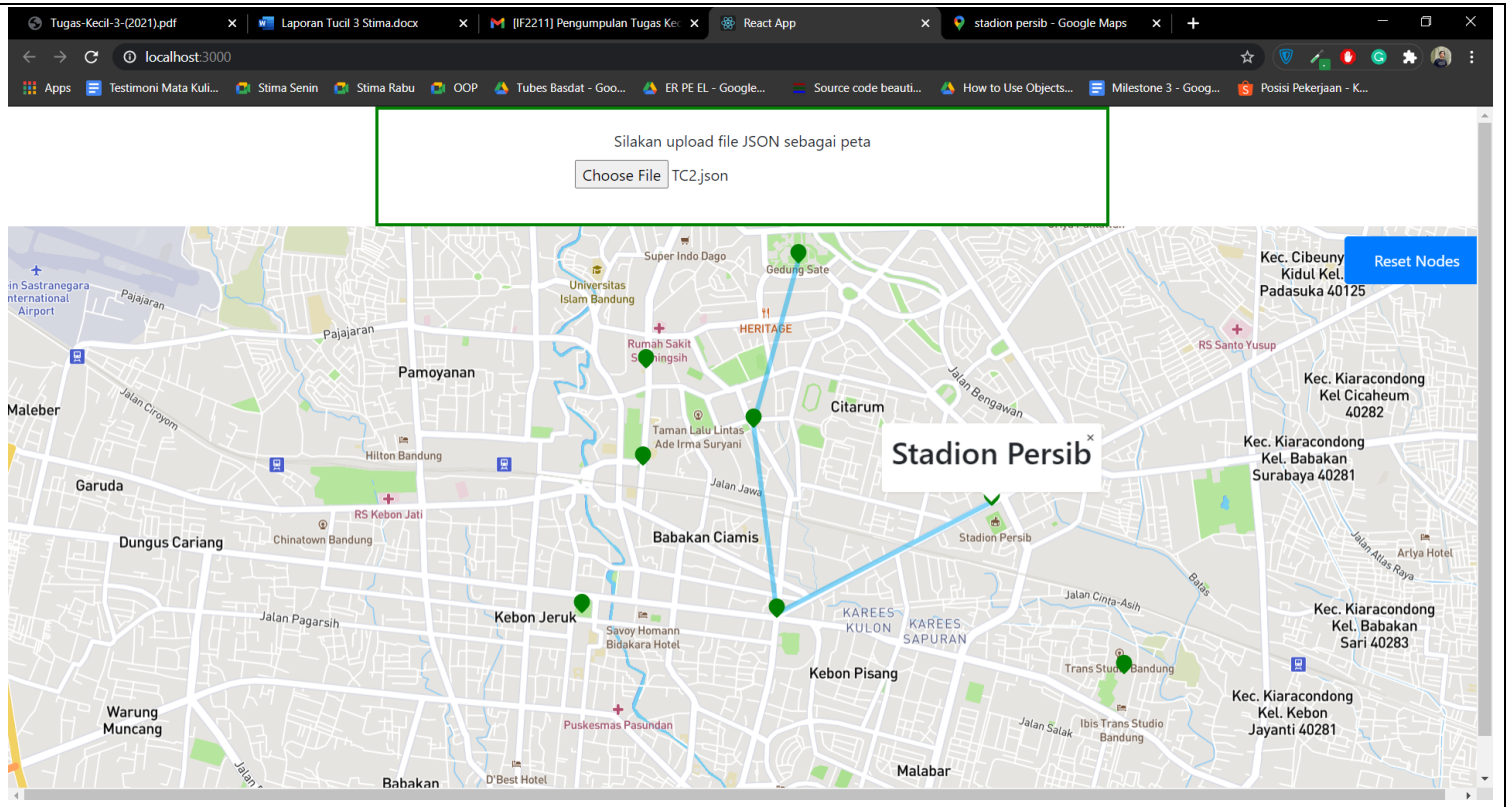

```
{
    "nodes":
        [
            {
                "id": 0,
                "name": "Taman Dago",
                "coordinate": [107.61244391401635, -6.898512260194929]
            },
            {
                "id": 1,
                "name": "Pertigaan Tamansari",
                "coordinate": [107.60845619489565, -6.8938593825294925]
            },
            {
                "id": 2,
                "name": "Pertigaan Dipati Ukur",
                "coordinate": [107.61520212611802, -6.887470051198127]
            },
```

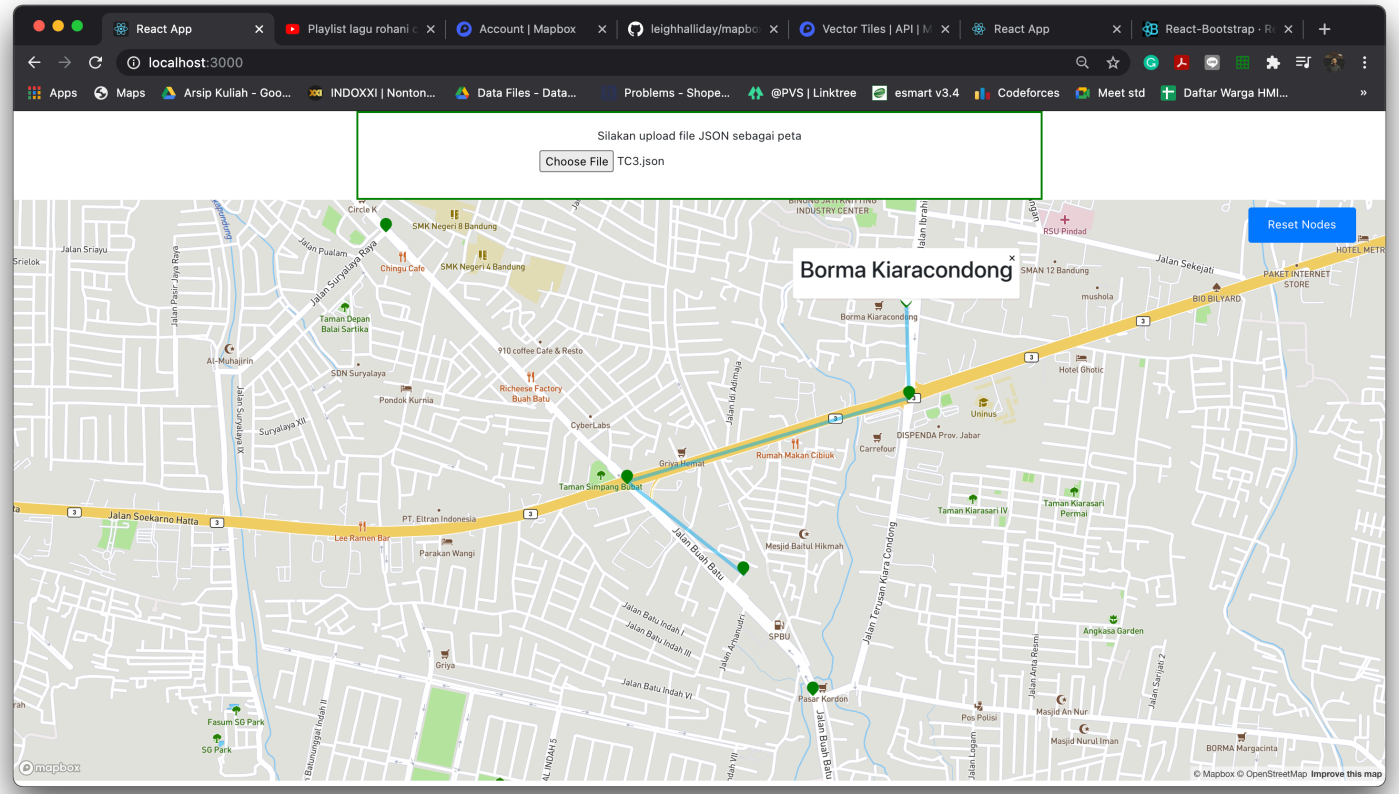
```
        {
            "id": 3,
            "name": "Ujung Tamansari Food Fest",
            "coordinate": [107.60828212400365, -6.887870730916851]
        },
        {
            "id": 4,
            "name": "Dago Suites",
            "coordinate": [107.60964411641272, -6.8832963104532014]
        },
        {
            "id": 5,
            "name": "Dago Asri",
            "coordinate": [107.61645443942624, -6.878626730536139]
        },
        {
            "id": 6,
            "name": "Pertigaan McD Dago",
            "coordinate": [107.61362362899371, -6.885192622300306]
        },
```

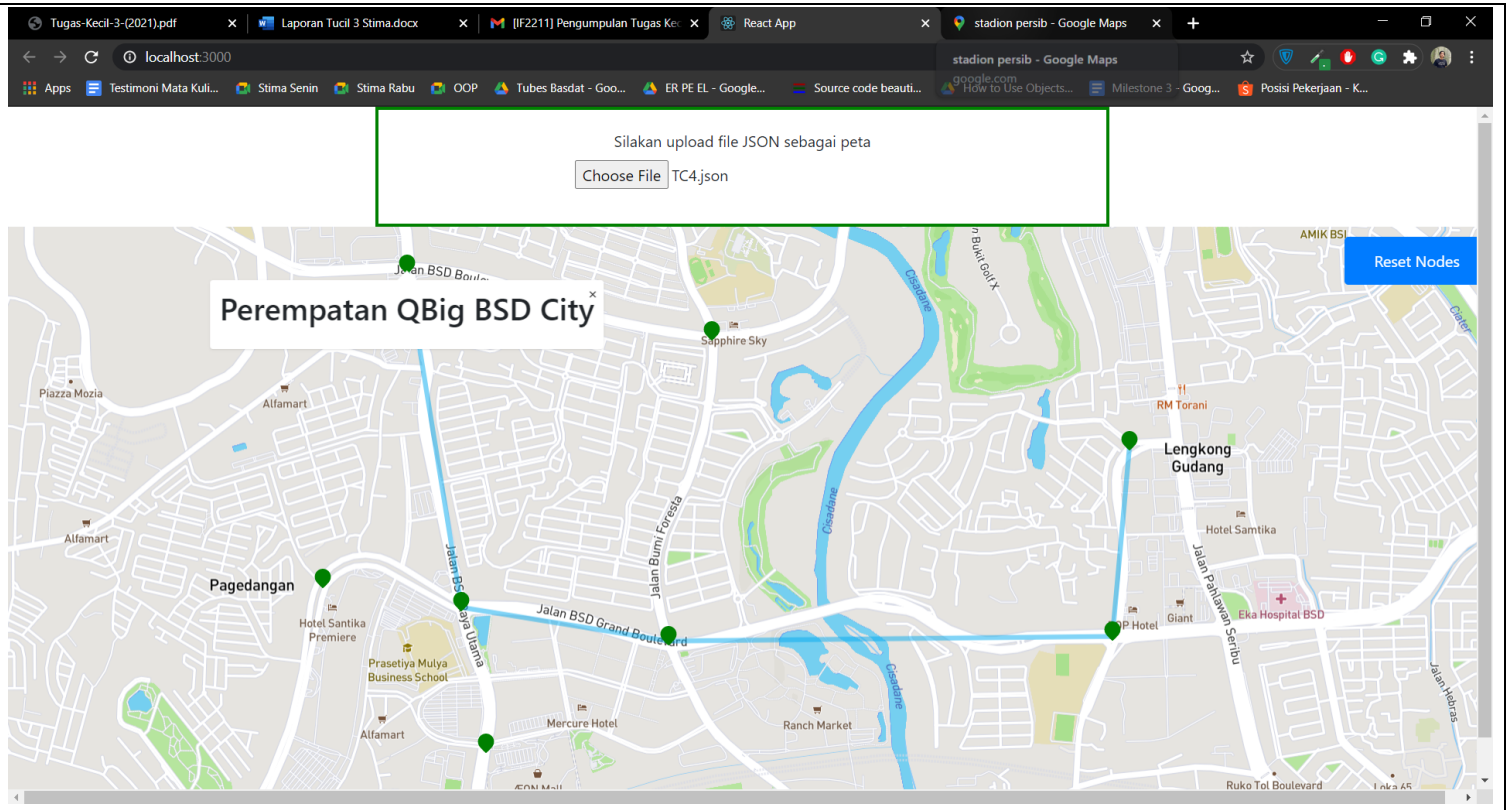
```
            {
                "id": 7,
                "name": "Pertigaan Ciumbuleuit",
                "coordinate": [107.60489623559742, -6.883315821249175]
            }
        ],

    "adjacency matrix":
        [
            [0, 0, 0, 0, 0, 0, 1, 0],
            [0, 0, 0, 1, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 1, 0],
            [0, 1, 0, 0, 0, 0, 1, 0],
            [0, 0, 0, 0, 0, 0, 1, 0],
            [0, 0, 0, 0, 0, 0, 1, 0],
            [1, 0, 1, 1, 1, 1, 0, 1],
            [0, 0, 0, 0, 0, 0, 1, 0]
        ]
}
```

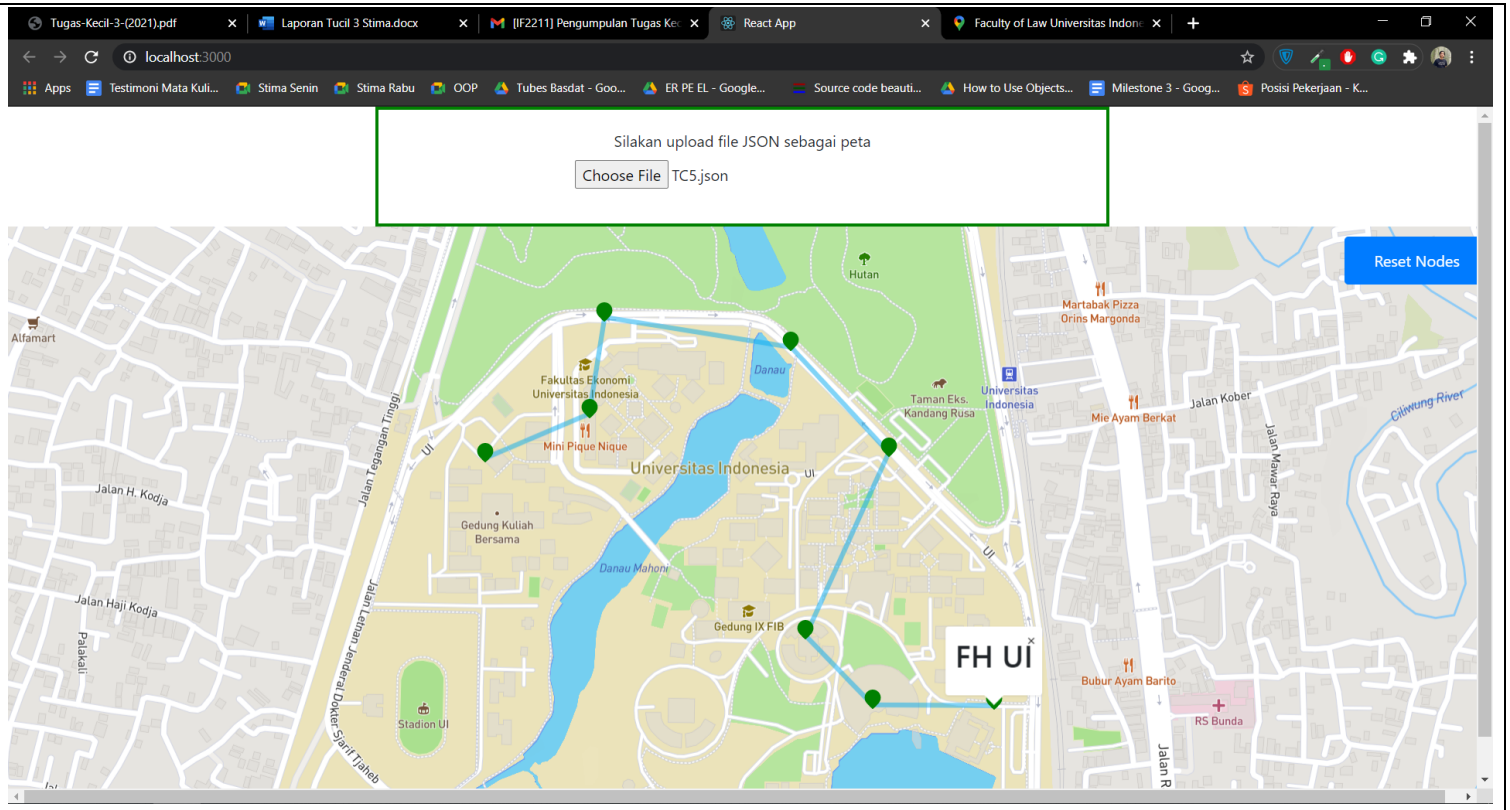

3. Input & Output

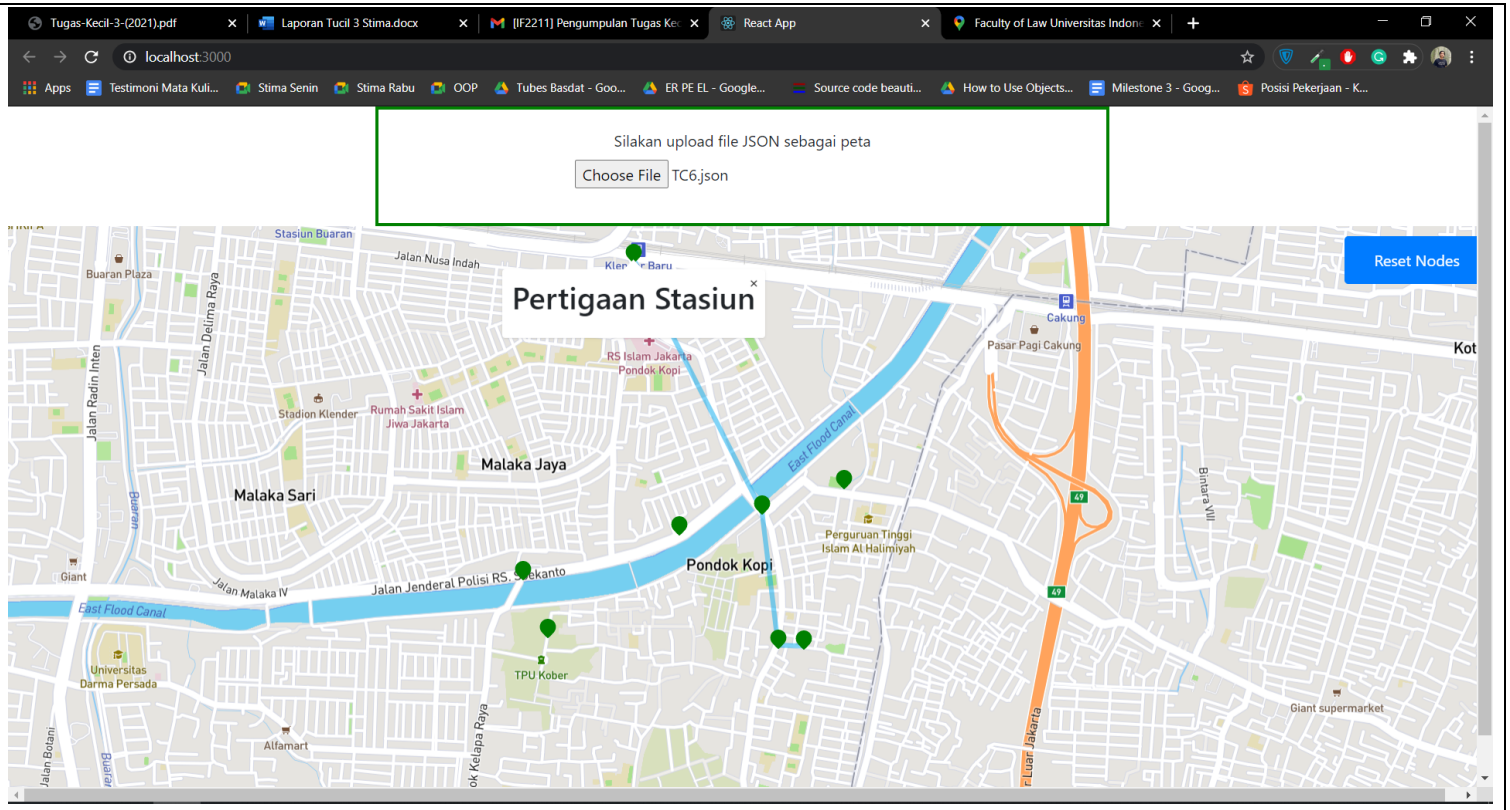
Isi file eksternal (.json)	Output
File json: TC1.json Rute: Pertigaan Ciumbuleuit menuju Ganyang	

File json: TC2.json Rute: Gedung Sate menuju Stadion Persib	

File json: TC3.json Rute: Borma Kiaracondong menuju Graha Merah Putih	

File json: TC4.json Rute: Perumahan Virginia Lagoon ke QBig BSD City	

File json: TC5.json Rute: FT UI ke FH UI	

File json: TC6.json Rute: Kelurahan Pondok Kopi ke Pertigaan Stasiun	

4. Alamat Github Kode Program

Berikut adalah pranala akses menuju repositori dari kode program ini:
https://github.com/andresjerriels/Tucil3Stima/tree/dev

5. *Check List* Program

Poin	Ya	Tidak
1. Program dapat menerima input graf	✓	
2. Program dapat menghitung lintasan terpendek	✓	
3. Program dapat menampilkan lintasan terpendek serta jaraknya	✓	
4. Bonus: Program dapat menerima input peta dengan Google Map API dan menampilkan peta	✓	