LAPORAN HASIL TUGAS KECIL 3 IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2020/2021
IMPLEMENTASI ALGORITMA A* UNTUK MENENTUKAN LINTASAN TERPENDEK

Disusun oleh :

Christoper Chandrasaputra	**13519074**
Billy Julius	**13519094**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2021

DAFTAR ISI

BAB I KODE PROGRAM

1.1 Kelas Node

```
from location import *

class Node :
    def __init__(self, name, location) :
        self.name = name
        self.location = location
        self.adjacentNode = []

    # add adjacent node name
    # input
    #   string : nodeName
    # I.S. : self.adjacentNode == [..]
    # F.S. : self.adjacentNode == [..,nodeName]
    def addAdjNode(self,nodeName):
        self.adjacentNode.append(nodeName)
```

1.2 Kelas Location

```
import math

class Location :
    def __init__(self, x, y) :
        self.x = x
        self.y = y

    # calculate euclidean distance of current location to other location
    # input
    #   Location : otherLocation
    # output : float
    def euclideanDist(self,otherLocation):
        return (math.sqrt((self.x-otherLocation.x)**2 +
(self.y-otherLocation.y)**2))


if __name__ == '__main__':
    l1 = Location(1,1)
```

```
    l2 = Location(2,1)

    x = l1.euclideanDist(l2)

    print(x)
```

1.3 Kelas PriorityQueue

```
PRIORITY_INDEX = 0

class PriorityQueue :
    def __init__(self):
        self.queue = []

    def __len__(self):
        return len(self.queue)

    # enqueue tuple to self.queue
    # input
    #   tuple : tup
    # I.S. : [..]
    # F.S. : [..,tup]
    # !! IMPORTANT : tup[0] must be a number as a priority that can be
compared !!
    def enqueue(self, tup):
        if len(self.queue) == 0:
            self.queue.append(tup)
        else:
            for i in range(len(self.queue)):
                if self.queue[i][PRIORITY_INDEX] >= tup[0]:
                    self.queue.insert(i,tup)
                    return
            self.queue.insert(len(self.queue),tup)

    # dequeue self.queue
    # I.S. : [front,..]
    # F.S. : [..]
    # output : tuple
    def dequeue(self):
        if len(self.queue) == 0:
```

```
            return []
        tup = self.queue[0]
        self.queue = self.queue[1:]
        return tup

if __name__ == '__main__':
    prioQueue = PriorityQueue()

    tup1 = (1,"A")
    tup2 = (3,"C")
    tup3 = (2,"D")
    tup4 = (1,"B")

    prioQueue.enqueue(tup1)
    prioQueue.enqueue(tup2)
    prioQueue.enqueue(tup3)
    prioQueue.enqueue(tup4)

    print(prioQueue.queue)
```

1.4 Kelas Graph

```
import math
from typing import overload
from location import *
from node import *
from prioqueue import *

EARTH_CIRCUMFERENCE = 40007863

class Graph :
    def __init__(self) :
        self.nodes = []

    # add a node to the graph
    # input
    #   Node : node
    # I.S. : nodes == [..]
    # F.S. : nodes == [..,node]
    def addNode(self, node) :
```

```
        self.nodes.append(node)

    # convert graph to array
    # output : [[[nodeName, location],..],[[[edgeSrcLoc.x, edgeSrcLoc.y],
[edgeTrgLoc.x, edgeTrgLoc.y]],..]]]
    def toArray(self):
        nodeList = []
        edgeList = []
        for node in self.nodes:

            tempNodeArr = [node.name, node.location.x, node.location.y]
            nodeList.append(tempNodeArr)

            adjNode = node.adjacentNode.copy()

            tempNodeLoc = node.location
            for adjNodeName in adjNode:
                tempAdjLoc = self.getNodeLoc(adjNodeName)
                if ([[tempNodeLoc.x,
tempNodeLoc.y],[tempAdjLoc.x,tempAdjLoc.y]] not in edgeList) and
([[tempAdjLoc.x,tempAdjLoc.y],[tempNodeLoc.x, tempNodeLoc.y]] not in
edgeList):
                    edgeList.append([[tempNodeLoc.x, tempNodeLoc.y],
[tempAdjLoc.x,tempAdjLoc.y]])

        return [nodeList,edgeList]

    # get node with the name of nodeName
    # input
    #   nodeName : string
    # output : Location
    def getNode(self, nodeName):
        for node in self.nodes:
            if node.name == nodeName:
                return node
        return None

    # get node location of a node with the name of nodeName
    # input
    #   nodeName : string
```

```
    # output : Location
    def getNodeLoc(self, nodeName):
        if self.getNode(nodeName) == None:
            return None
        return self.getNode(nodeName).location

    # calculate Euclidean Distance of
    # input
    #   string : nodeName
    # output : float
    def calculateDistance(self, srcNodeName, trgNodeName):
        return
self.getNodeLoc(srcNodeName).euclideanDist(self.getNodeLoc(trgNodeName))

    # get total cost from srcNodeName to trgNodeName with the goal
goalNodeName
    # input
    #   string : srcNodeName, trgNodeName, goalNodeName
    # output : float
    def calculateTotalCost(self, srcNodeName, trgNodeName, goalNodeName):
        return  self.calculateDistance(srcNodeName,trgNodeName) +
self.calculateDistance(trgNodeName,goalNodeName)

    # fill the graph from an input file
    # input
    #   string : fileName
    # I.S. : self.nodes == [..]
    # F.S. : self.nodes == [..,newnode1,newnode2,..,newnodeN]
    def fillGraphWithFile(self, fileName) :
        self.nodes = []
        f = open(fileName,"r")
        readOut = f.readlines()

        nodeCount = int(readOut[0].replace('\n',''))

        # fill the nodes
        for i in range (1,nodeCount+1) :
            line = readOut[i].replace('\n','').split(' ')
            Loc = Location(float(line[1]) , float(line[2]))
            tempNode = Node(line[0], Loc)
```

```
            self.addNode(tempNode)

        # fill the edges
        i = 0
        for lineIdx in range(nodeCount+1, nodeCount*2+1) :
            line = readOut[lineIdx].replace('\n','').split(' ')
            for j in range(nodeCount):
                if j != i and line[j]=='1':
                    self.nodes[i].addAdjNode(self.nodes[j].name)
            i += 1
        f.close()

    # get total cost from srcNodeName to trgNodeName with the goal
goalNodeName
    # input
    #   string : srcNodeName, goalNodeName
    # output : array of string, float, boolean
    def AStar(self, srcNodeName, goalNodeName):
        ## Guard
        if self.getNode(srcNodeName) == None or self.getNode(goalNodeName)
== None:
            return None, None, False
        ## PriorityQueue elements as tuple index
        TOTALCOST_INDEX = 0
        CURRENTNODENAME_INDEX = 1
        VISITEDNODE_INDEX = 2

        ## PriorityQueue elements : (cost,currentNodeName,visitedNode)
        queue = PriorityQueue()

        ## PriorityQueue initialization (using start node)
        temptup =
(self.calculateDistance(srcNodeName,goalNodeName),srcNodeName,[srcNodeName
])
        queue.enqueue(temptup)

        while len(queue)>0 and queue.queue[0][CURRENTNODENAME_INDEX] !=
goalNodeName:
            currentNode = queue.dequeue() #
(cost,currentNodeName,visitedNode) - (float, string, [string])
```

```
            currentAccCost = currentNode[TOTALCOST_INDEX]
            print("CURRENT COST  ", currentAccCost)
            node = self.getNode(currentNode[CURRENTNODENAME_INDEX])

            currentAccCost -=
self.calculateDistance(node.name,goalNodeName)

            visitedNode = currentNode[VISITEDNODE_INDEX].copy()

            adjNodeNames = node.adjacentNode

            for adjNodeName in adjNodeNames:

                if (adjNodeName not in visitedNode):
                    ## create priorityQueue object to be queued into
PriorityQueue queue
                    # create total cost
                    tempAccCost = currentAccCost +
self.calculateTotalCost(node.name,adjNodeName,goalNodeName)

                    # create path(visited node)
                    tempVisitedNode = visitedNode.copy()
                    tempVisitedNode.append(adjNodeName)

                    # create temporary tuple
                    temptup = (tempAccCost,adjNodeName,tempVisitedNode)

                    queue.enqueue(temptup)

        pathNames = []
        dist = 0.0
        success = True

        if len(queue)>0: # found path from Source Node to Goal Node
            print("COST = ",end='')
            print(queue.queue[0][TOTALCOST_INDEX])
            dist = queue.queue[0][TOTALCOST_INDEX]

            print("PATH = ",end='')
```

```
            print(queue.queue[0][VISITEDNODE_INDEX][0],end='')
            pathNames.append(queue.queue[0][VISITEDNODE_INDEX][0])
            for nodeName in queue.queue[0][VISITEDNODE_INDEX][1:]:
                print('-'+nodeName,end='')
                pathNames.append(nodeName)
            print()

        else: # no path found from Source Node to Goal Node
            print("NO PATH FOUND")
            success = False

        return pathNames, (dist/360) * EARTH_CIRCUMFERENCE, success
```

1.5 Main Program

```
from graph import *
from flask import Flask , render_template , request
# import matplotlib.pyplot as plt
# import networkx as nx
HTML_TEMPLATE = "map.html"

def printDefaultTemplate(graphInfo=[]):
    return render_template(HTML_TEMPLATE, route = "", coords = [], dist =
0 , graphInfo = graphInfo)

app = Flask(__name__)
app.config['SECRET_KEY'] = 'thisisjustsomesecretkey'

G = Graph()
gInfo = []
@app.route('/', methods=['GET', 'POST'])
def main() :
    return printDefaultTemplate()

@app.route('/load-file/', methods=['GET', 'POST'])
def loadFile() :
    fileNamePath = request.form['test']
    if (len(fileNamePath)==0):
        return printDefaultTemplate()
    print("SELECTED FILE =",fileNamePath)
```

```
    G.fillGraphWithFile("../test/" + fileNamePath)
    gInfo = G.toArray()
    return printDefaultTemplate(gInfo)

@app.route('/calculate-route/', methods=['GET', 'POST'])
def calculateRoute():
    startNode = request.form.get("startNodeName")
    goalNode = request.form.get("goalNodeName")
    if startNode==None or goalNode==None:
        return printDefaultTemplate()
    pathNames, pathDist, pathSuccess = G.AStar(startNode,goalNode)
    gInfo = G.toArray()
    if (pathSuccess) :
        pathCoords = []
        for i in range(len(pathNames)) :
            currLoc = [G.getNodeLoc(pathNames[i]).x,
G.getNodeLoc(pathNames[i]).y]
            pathCoords.append(currLoc)
        routeResult = '-'.join(pathNames)
        return render_template(HTML_TEMPLATE, route = routeResult, coords
= pathCoords, dist = pathDist , graphInfo = gInfo)
    else:
        badRouteResult = "No path found from " + startNode + " to " +
goalNode
        return render_template(HTML_TEMPLATE, route = badRouteResult,
coords = [], dist = 0 , graphInfo = gInfo)

if __name__ == '__main__':
    app.run(debug=True)
```

1.6 File layout.html

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
```

```
        <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min
.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoR
xT2MZw1T" crossorigin="anonymous">
        <link rel="stylesheet" type="text/css" href="{{ url_for('static' ,
filename = 'main.css') }}">

        <!-- Here API -->
        <meta name="viewport" content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=yes">
        <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
        <link rel="stylesheet" type="text/css"
href="https://js.api.here.com/v3/3.1/mapsjs-ui.css" />
        <script type="text/javascript"
src="https://js.api.here.com/v3/3.1/mapsjs-core.js"></script>
        <script type="text/javascript"
src="https://js.api.here.com/v3/3.1/mapsjs-service.js"></script>
        <script type="text/javascript"
src="https://js.api.here.com/v3/3.1/mapsjs-ui.js"></script>
        <script type="text/javascript"
src="https://js.api.here.com/v3/3.1/mapsjs-mapevents.js"></script>

        <style>
          #term-table {
            font-family: Arial, Helvetica, sans-serif;
            border-collapse: collapse;
            width: 100%;
          }

          #term-table td, #term-table th {
            border: 1px solid #ddd;
            padding: 8px;
          }

          #term-table tr:nth-child(even){background-color: #f2f2f2;}

          #term-table tr:hover {background-color: #ddd;}
```

```
          #term-table th {
            padding-top: 12px;
            padding-bottom: 12px;
            text-align: left;
            background-color: #4d4d4d;
            color: white;
          }
        </style>

        <title>A* Path</title>
    </head>

    <body>
        <header class="site-header">
            <nav class="navbar navbar-expand-md navbar-dark bg-steel
fixed-top">
              <div class="container">
                <a class="navbar-brand mr-4" href="/">A* Path</a>
              </div>
            </nav>
        </header>

        <main role="main" class="container">
          <br><br>
          <div class="row">
            <div class="col-md-8">
              {% block content %}{% endblock %}
            </div>
            {% block sideInfo %}{% endblock %}
          </div>
        </main>

        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE
1Pi6jizo" crossorigin="anonymous"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.mi
n.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86d
IHNDz0W1" crossorigin="anonymous"></script>
```

```
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.j
s"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xI
M+B07jRM" crossorigin="anonymous"></script>
    </body>
</html>
```

1.7 File map.html

```
{% extends "layout.html" %}
{% block content %}
<div style="width: 640px; height: 480px" id="mapContainer">
    <script>
        // add a marker
        function addMarkerToGroup(group, coordinate, html) {
            var marker = new H.map.Marker(coordinate);
            // add custom data to the marker
            marker.setData(html);
            group.addObject(marker);
        }

        // add an info bubble for each marker
        function addInfoBubble(map){
            // add the group to the map
            map.addObject(group);

            // add 'tap' event listener, that opens info bubble, to the
group
            group.addEventListener('tap', function (evt) {
                // event target is the marker itself, group is a parent
event target
                // for all objects that it contains
                var bubble =  new
H.ui.InfoBubble(evt.target.getGeometry(), {
                // read custom data
                content: evt.target.getData()
                });
                // show info bubble
                ui.addBubble(bubble);
```

```
            }, false);
        }

        // add a path result from algorithm to the map
        function addResultPath(map) {
            var lineString = new H.geo.LineString();
            for (let i = 0; i<points.length; i++){
                lineString.pushPoint({lat:points[i][0],
lng:points[i][1]});
            }

            // draw the line
            map.addObject(new H.map.Polyline(
                lineString, { style: { lineWidth: 3, strokeColor:
"rgba(33, 255, 0, 1.0)" }}
            ));
        }

        // add graph to map
        function addGraphToMap(map){
            var nodesInfo = graph[0]
            // var lineString = new H.geo.LineString();
            // ADD NODES TO MAP
            for (let i = 0; i<nodesInfo?.length; i++){
                // add a marker to the group
                addMarkerToGroup(group, {lat:nodesInfo[i][1],
lng:nodesInfo[i][2]}, nodesInfo[i][0]);
            }
            // // ADD EDGES TO MAP
            var edgesInfo = graph[1]
            for (let i = 0; i<edgesInfo?.length; i++){
                var lineString = new H.geo.LineString();
                lineString.pushPoint({lat:edgesInfo[i][0][0],
lng:edgesInfo[i][0][1]}); // add src node location
                lineString.pushPoint({lat:edgesInfo[i][1][0],
lng:edgesInfo[i][1][1]}); // add trg node location
                map.addObject(new H.map.Polyline(
                    lineString, { style: { lineWidth: 4 , strokeColor:
"rgba(0, 113, 255, 1.0)" }}
                ));
```

```
        }
        // ADD INFO BUBBLE
        addInfoBubble(map);
    }

    function clearMap(){
        map.removeObjects(map.getObjects());
    }

    function clearPath(){
        clearMap();
        addGraphToMap(map);
    }

    // receive the route string
    var route = JSON.parse('{{ route|tojson }}');
    // receive the points pass
    var points = JSON.parse('{{ coords|tojson }}');
    // receive the graph data pass
    var graph = JSON.parse('{{ graphInfo|tojson }}');

    // // other value initialization
    // var pointCount = 0;

    // create a new group
    var group = new H.map.Group();

    // init the platform
    var platform = new H.service.Platform({
        'apikey': 'ktOMjYT0KrFXrOWL7Sxp475UB20KgVQtBpAnsTmgnTA'
    });

    // create a layer
    var layer = platform.createDefaultLayers();

    // search camera position
    var avgLat = 0.0;
    var avgLng = 0.0;
    if (points.length>0){
        var sumLat = 0.0;
```

```
            var sumLng = 0.0;
            for (let i=0; i<points.length; i++){
                sumLat += points[i][0]; sumLng += points[i][1];
            }
            avgLat = sumLat/points.length; avgLng = sumLng/points.length;
        } else if (graph.length>0){
            var sumLat = 0.0;
            var sumLng = 0.0;
            var nodeInfo = graph[0];
            for (let i=0; i<nodeInfo.length; i++){
                sumLat += nodeInfo[i][1]; sumLng += nodeInfo[i][2];
            }
            avgLat = sumLat/nodeInfo.length; avgLng =
sumLng/nodeInfo.length;
        }

        // instantiate (and display) a map object:
        var map = new H.Map(
            document.getElementById('mapContainer'),
            layer.vector.normal.map,
            {
                zoom: 15,
                center: { lat: avgLat, lng: avgLng },
                pixelRatio: window.devicePixelRatio || 1
            });

        // add a resize listener to make sure that the map occupies the
whole container
        window.addEventListener('resize', () =>
map.getViewPort().resize());

        // MapEvents enables the event system
        // Behavior implements default interactions for pan/zoom (also on
mobile touch environments)
        var behavior = new H.mapevents.Behavior(new
H.mapevents.MapEvents(map));

        // Create the default UI components
        var ui = H.ui.UI.createDefault(map, layer);
```

```
        clearMap();
        addGraphToMap(map);
        if (points.length>0) addResultPath(map);
    </script>
</div>

{% endblock content %}

{% block sideInfo %}
<div class="col-md-4">
    <div class="content-section">
      <h3>Navigation</h3>
      <p class='text-muted'>Kolom ini digunakan untuk mengisi path file
eksternal, lokasi asal, dan lokasi tujuan
        <article class="media content-section">
            <div class="media-body">
                <br> <!-----------------Node Option-------------------->
                <div>
                    <form action="/load-file/" method="POST">
                        <input type="text" name="test">
                        <input type="submit" value="Use File">
                    </form>
                </div>
                <br> <!-----------------Node Option-------------------->
                <div>
                    <form action="/calculate-route/" method="POST">
                        <div>
                            <label for="startNodeName">Lokasi asal
:</label>
                            <select id="startNodeName" name =
"startNodeName">
                                {% for nodeName in graphInfo[0] %}
                                    <option
value={{nodeName[0]}}>{{nodeName[0]}}</option>
                                {% endfor %}
                            </select>
                        </div>

                        <div>
```

```
                            <label for="goalNodeName">Lokasi tujuan
:</label>
                            <select id="goalNodeName" name =
"goalNodeName">
                                {% for nodeName in graphInfo[0] %}
                                    <option
value={{nodeName[0]}}>{{nodeName[0]}}</option>
                                {% endfor %}
                            </select>
                        </div>

                        <input type="submit" value="Find Route">
                    </form>

                </div>
                <br>
                <div>
                    <!-- <p>Total cost : {{dist}}</p>
                    <p>Total cost : {{dist}}</p> -->
                    <p>Route : {{route}}</p>
                    <p>Total distance : {{dist}} m </p>
                </div>

                <br>
<!-------------------------------------------------------------------->
                <button onclick="clearPath()"> Clear generated path
</button>
            </div>
        </article>
      </p>
    </div>
</div>
{% endblock sideInfo %}
```

1.8 File main.css

```
body {
    background: #4b4b4b;
    color: #333333;
    margin-top: 5rem;
```

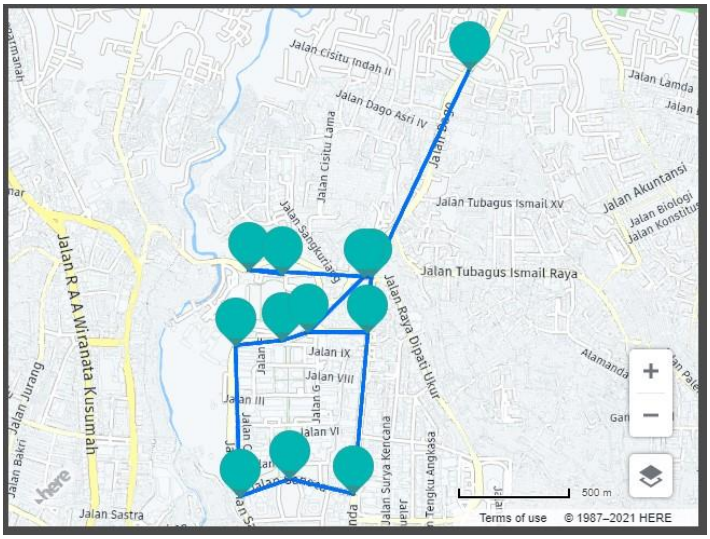
```
}

h1, h2, h3, h4, h5, h6 {
  color: #444444;
}

.bg-steel {
  background-color: #131414;
}

.site-header .navbar-nav .nav-link {
  color: #cbd5db;
}

.site-header .navbar-nav .nav-link:hover {
  color: #ffffff;
}

.site-header .navbar-nav .nav-link.active {
  font-weight: 500;
}

.content-section {
  background: #ffffff;
  padding: 10px 20px;
  border: 1px solid #dddddd;
  border-radius: 3px;
  margin-bottom: 20px;
}

.article-title {
  color: #444444;
}

a.article-title:hover {
  color: #428bca;
  text-decoration: none;
}

.article-content {
```

```
  white-space: pre-line;
}

.article-img {
  height: 65px;
  width: 65px;
  margin-right: 16px;
}

.article-metadata {
  padding-bottom: 1px;
  margin-bottom: 4px;
  border-bottom: 1px solid #e3e3e3
}

.article-metadata a:hover {
  color: #333;
  text-decoration: none;
}

.article-svg {
  width: 25px;
  height: 25px;
  vertical-align: middle;
}

.account-img {
  height: 125px;
  width: 125px;
  margin-right: 20px;
  margin-bottom: 16px;
}

.account-heading {
  font-size: 2.5rem;
}
```

BAB II PETA/GRAF INPUT

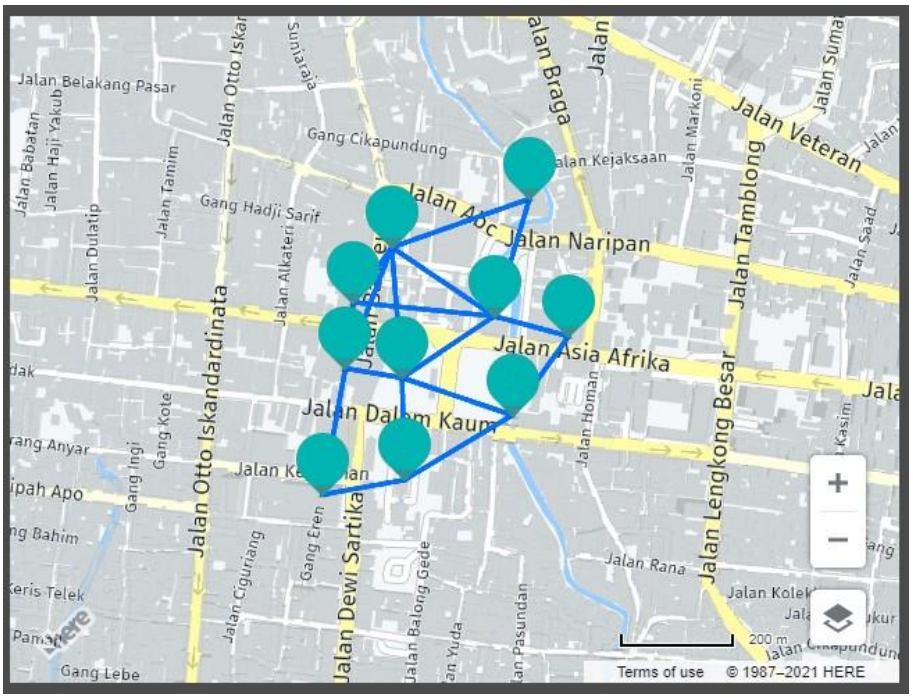
2.1 Peta/Graf Testcase 1

```
12
GerbangDepanITB -6.893173883546799 107.61044133458908
GerbangBelakangITB -6.8877080705653375 107.61015489128899
McDonaldDago -6.885188781827067 107.6134019790495
PintuMasukSabugaBakSil -6.885087592910358 107.61014041299113
BormaDago -6.876997641116672 107.61759464228754
GaneshaSimpangDago -6.8937694988394265 107.61297196057514
GaneshaSimpangTamansari -6.893868021598134 107.60844750807198
LengkunganTamansari -6.887898401045376 107.6082760931378
PerempatanDago -6.885207421878972 107.61368361101937
BabakanSiliwangi -6.884927820894696 107.60878857972264
DayangSumbi -6.8873829782301845 107.61117574562289
DayangSumbiSimpangDago -6.887388303937222 107.61354681823717
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 1 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0 0 1
0 0 1 1 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 1 0 1 0
```



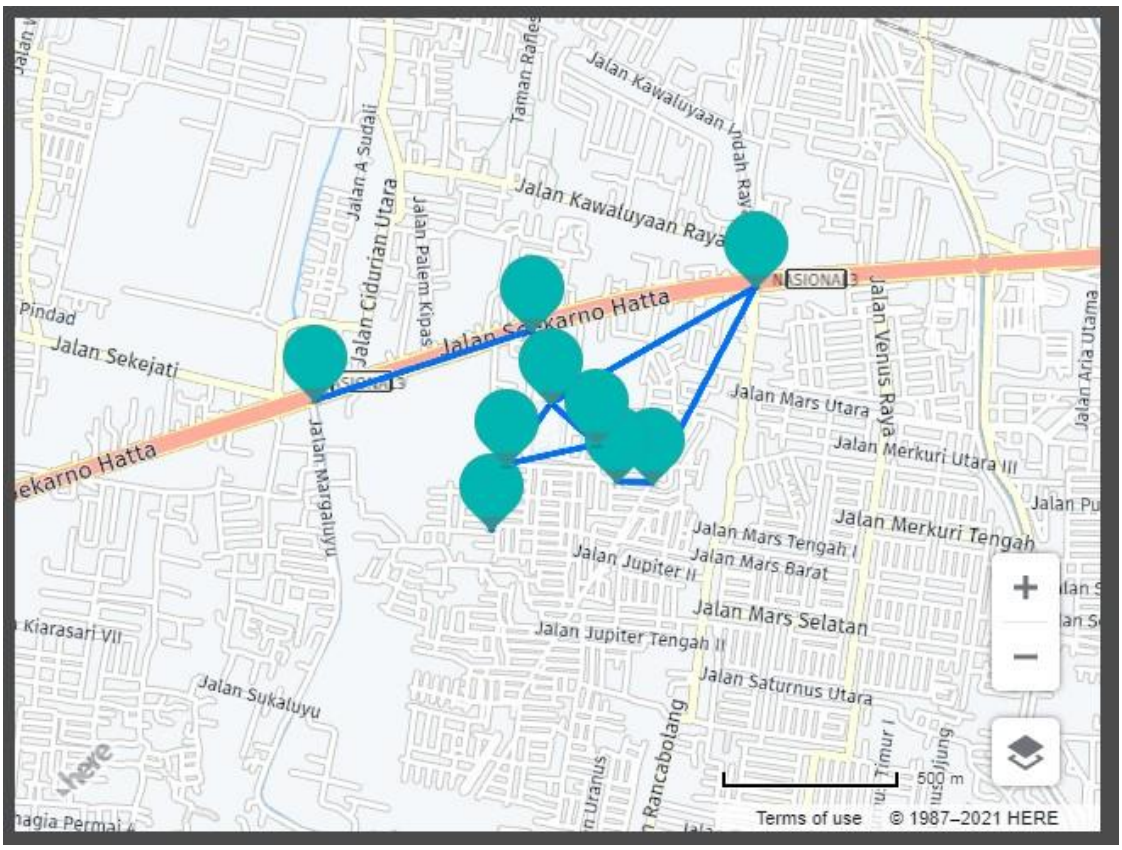
2.2 Peta/Graf Testcase 2

```
10
AlunAlunBandung -6.92182559249719 107.60695420607672
MuseumAsiaAfrika -6.921189604928431 107.60952504581432
PLNBandung -6.920880652884007 107.60837577208487
Cikapundung -6.919081870809035 107.60892135022254
KantorPos -6.920667768326371 107.60619855988753
MasjidRayaBandung -6.921681840605713 107.60607795650104
PendopoBandung -6.9233941110536215 107.60698555328437
BankPanin -6.919826236345531 107.60679326883204
GrandYogyaKepatihan -6.923618925963251 107.60572256674298
Cafe67 -6.922399953684318 107.60865480249316
0 0 1 0 0 1 1 1 0 1
0 0 1 0 0 0 0 0 0 1
1 1 0 1 1 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 1 0 0
1 0 0 0 1 0 0 1 1 0
1 0 0 0 0 0 0 0 1 1
1 0 1 1 1 1 0 0 0 0
0 0 0 0 0 1 1 0 0 0
1 1 0 0 0 0 1 0 0 0
```



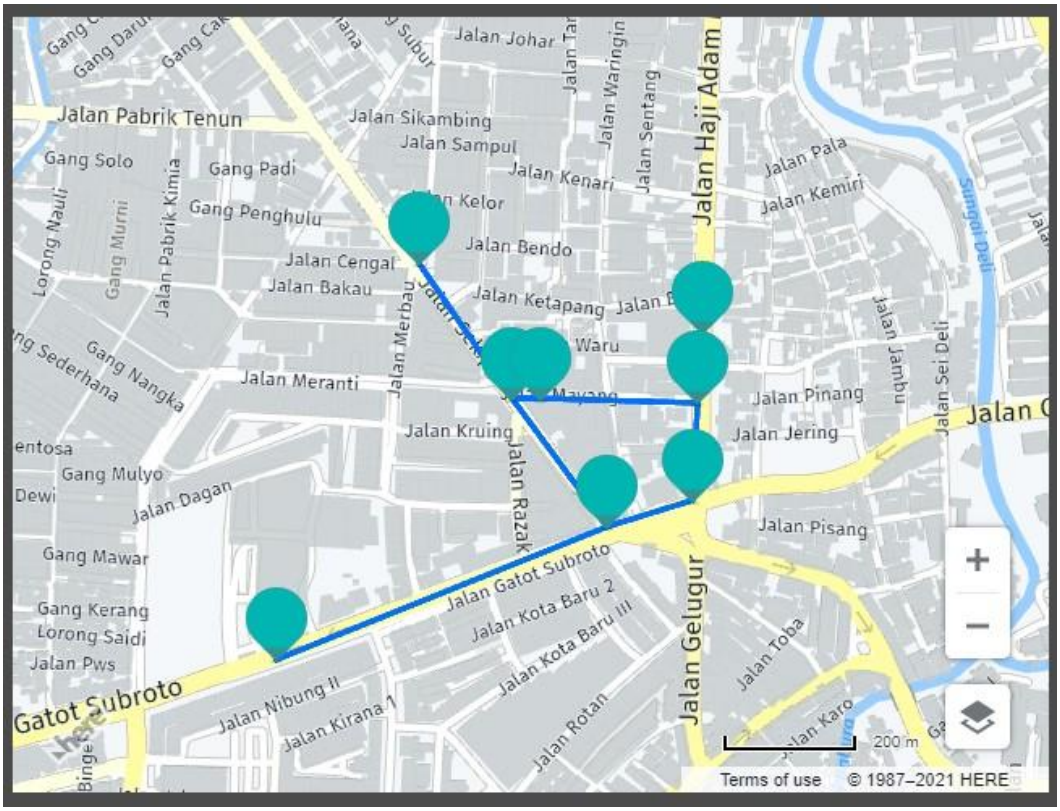
2.3 Peta/Graf Testcase 3

```
9
MetroIndahMall -6.942120717034387 107.65874719557186
JlSoekarnoHatta -6.940290411439278 107.65824797384171
JembatanSungaiCicadas -6.942040661767213 107.65277526794446
YamahaServiceCenter -6.943154221380146 107.65990485388869
JlJupiterBarat26 -6.943648609245794 107.65761911899006
TamanS2 -6.945294809444772 107.65724156585514
TamanJupiter -6.9440854448675 107.66042481163986
JlJupiterBaratUtama -6.944109165356709 107.66130417588958
JlRayaCirebonBandung -6.939199773655335 107.66391628016386
0 1 0 1 1 0 0 0 1
1 0 1 1 0 0 0 0 0
0 1 0 0 0 0 0 0 0
1 1 0 0 1 0 1 0 0
1 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 1 0 1
1 0 0 0 0 0 0 1 0
```



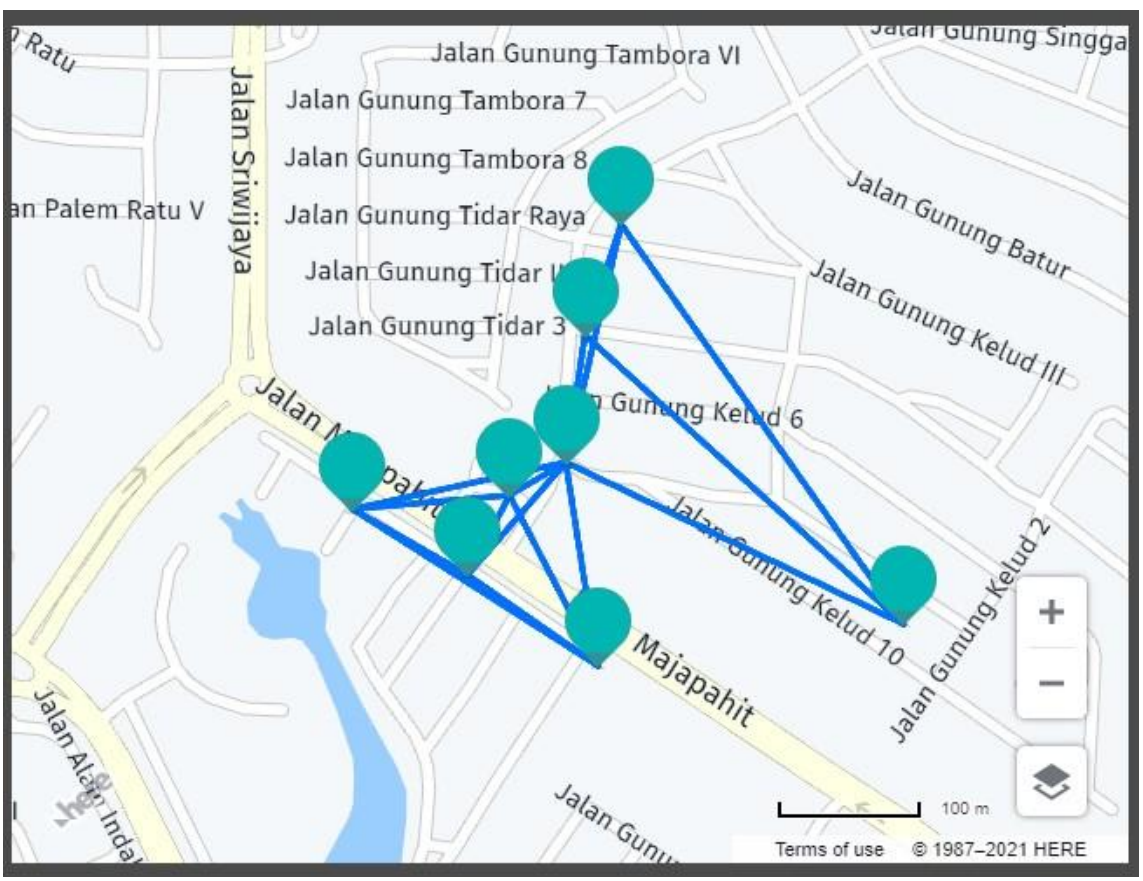
2.4 Peta/Graf Testcase 4

```
8
CarrefourMedanFair 3.5913 98.66331
SekipSimpangGatsu 3.59306 98.66769
PerempatanGatsu 3.59341 98.66882
AdamMalikSimpangMayang 3.5947 98.66889
PizzaHutAdamMalik 3.59562 98.66895
KalamKudusMedan 3.59476 98.6668
SekipSimpangMayang 3.59476 98.66643
UnpriSekip 3.59655 98.6652
0 1 0 0 0 0 0 0
1 0 1 0 0 0 1 0
0 1 0 1 0 0 0 0
0 0 1 0 1 1 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 1 0
0 1 0 0 0 1 0 1
0 0 0 0 0 0 1 0
```



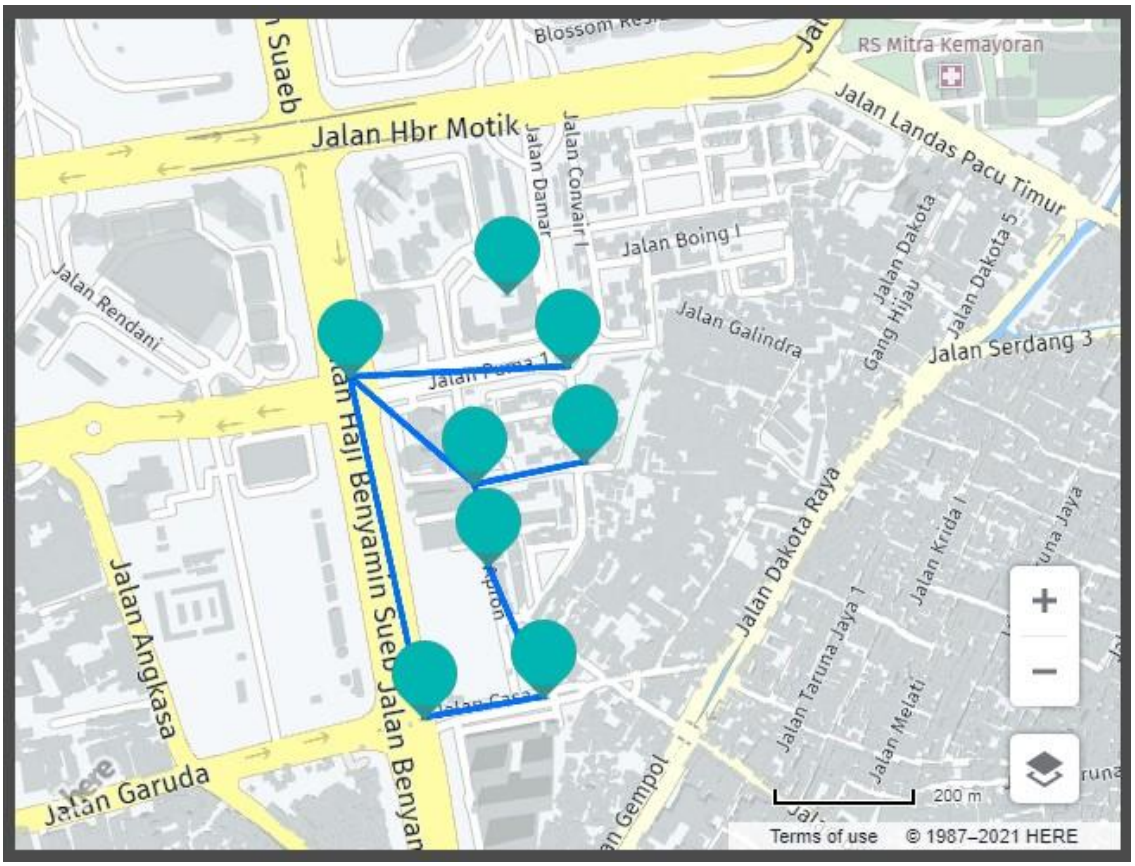
2.5 Peta/Graf Testcase 5

```
8
myHouse -6.3441000539837615 107.15762243971379
posSatpam -6.343113046621595 107.15554640995084
pomBensin -6.343314313629815 107.15520040499138
theHarvest -6.343801013744161 107.15494207962544
abubaSteak -6.343398480022214 107.15423665861792
jlnKeludRaya -6.342334156903663 107.15566451270803
redDoorz -6.341657014490506 107.15588706158087
indomaret -6.344354191621068 107.1557430069852
0 1 0 0 0 1 1 0
1 0 1 1 1 1 0 1
0 1 0 1 1 0 0 1
0 1 1 0 1 0 0 1
0 1 1 1 0 0 0 1
1 1 0 0 0 0 1 0
1 1 0 0 0 1 0 0
0 1 1 1 1 0 0 0
```



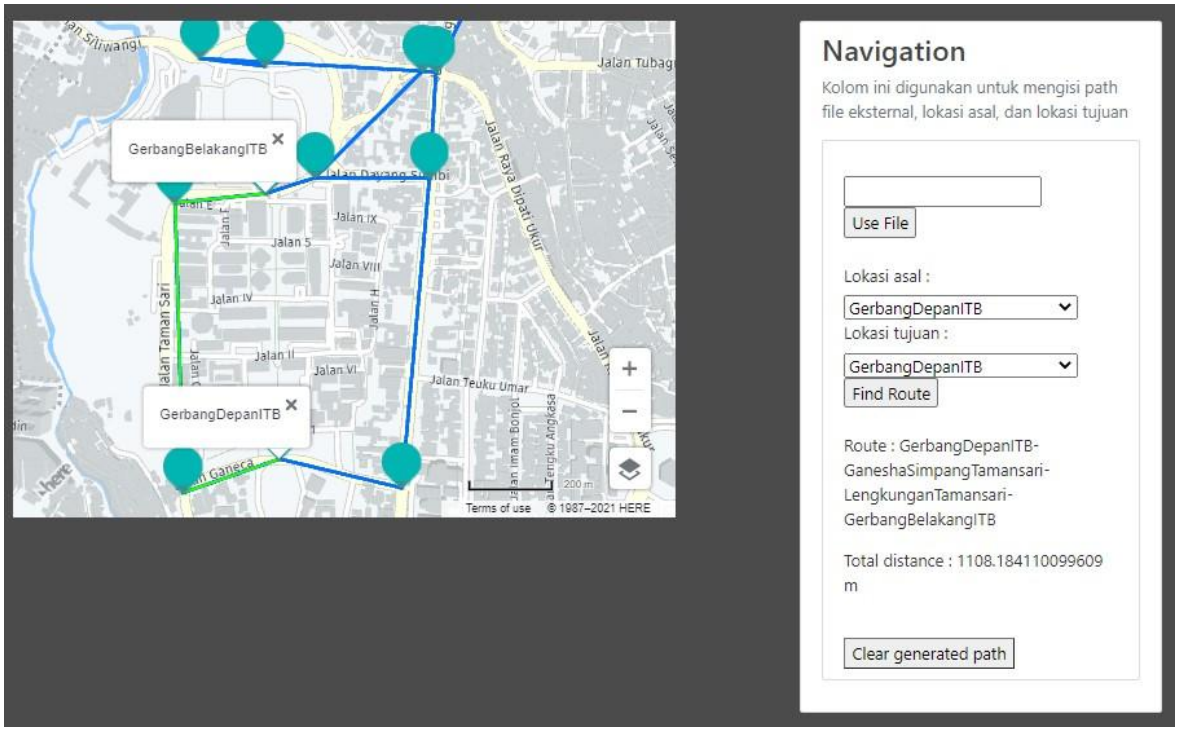
2.6 Peta/Graf Testcase 6

```
8
MasjidAkbar -6.155596454799754 106.85367280900618
HaltePumaRaya -6.1557251235936015 106.85096449092164
MonumenOndelOndel -6.159923146837551 106.85188711889913
GrandPalace -6.159665823160958 106.85337530710886
BaksoPakPur -6.158048377736383 106.85268204886337
ApartemenPuri -6.157055851616249 106.85251566823723
WarungWonokromo -6.15677095133923 106.85388369426994
MediteraniaResidence -6.15469399578285 106.8529316210351
0 1 0 0 0 0 0 0
1 0 1 0 0 1 0 0
0 1 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 1 0 0
0 1 0 0 1 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
```

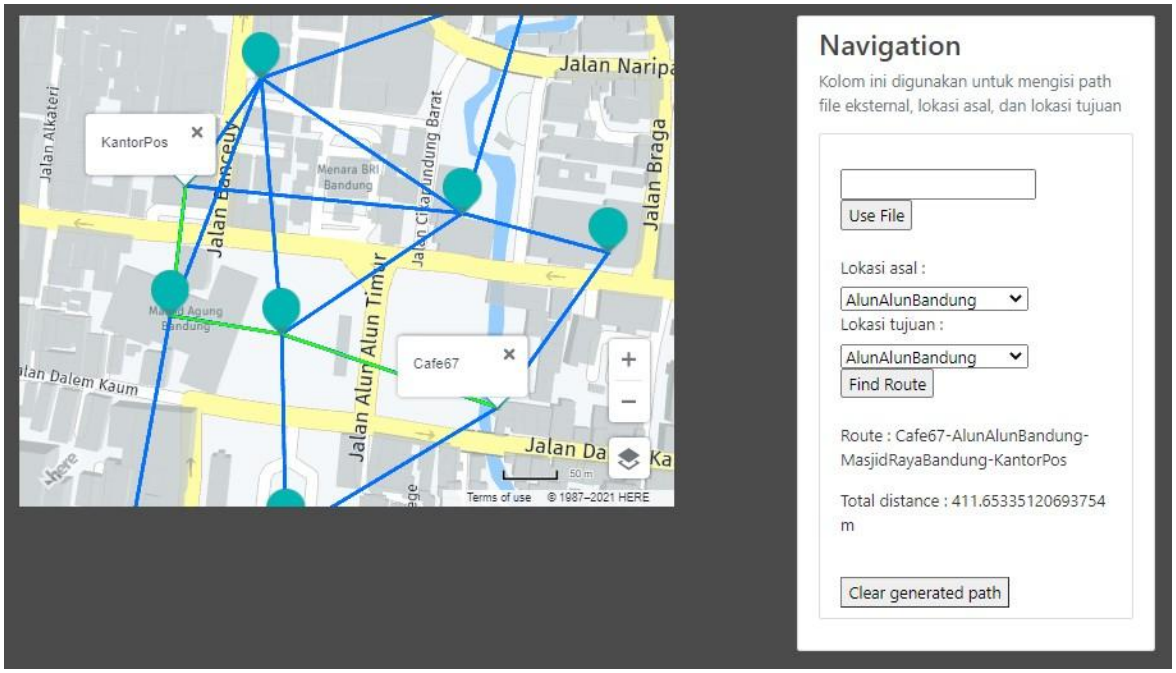


BAB III *SCREENSHOT* PETA YANG MENUNJUKKAN LINTASAN TERPENDEK SEPASANG SIMPUL

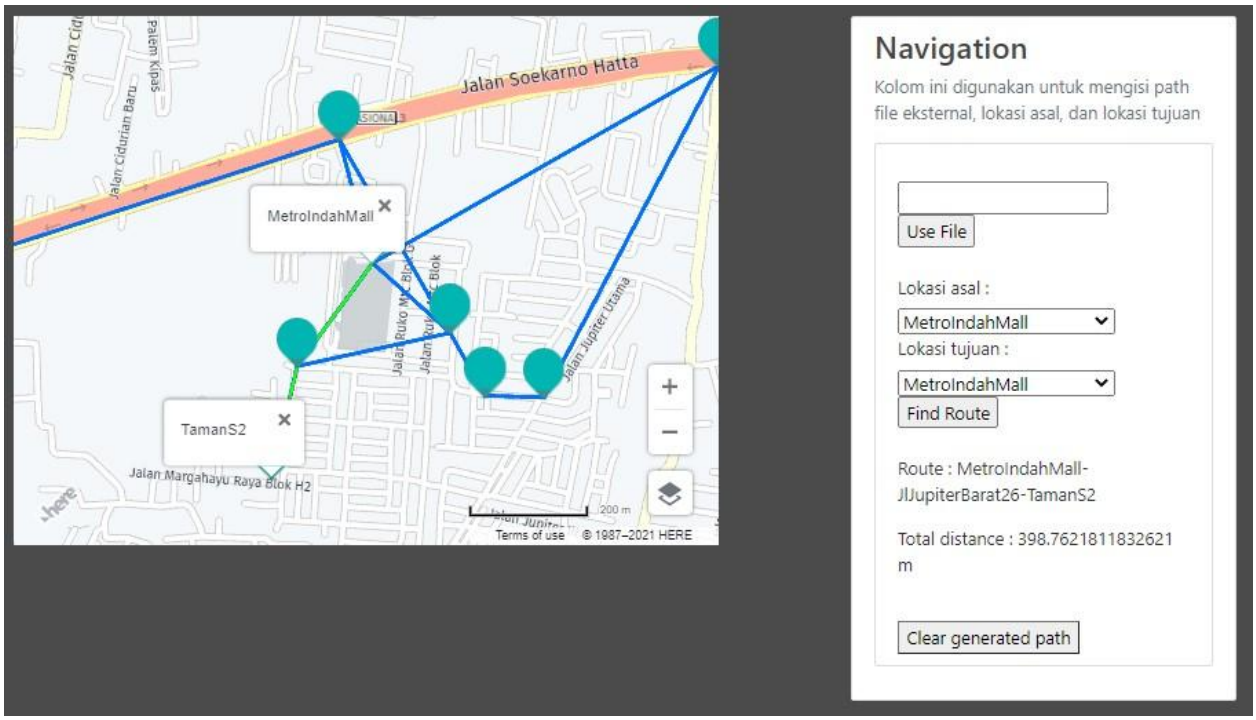
3.1 Testcase 1 : GerbangDepanITB menuju GerbangBelakangITB



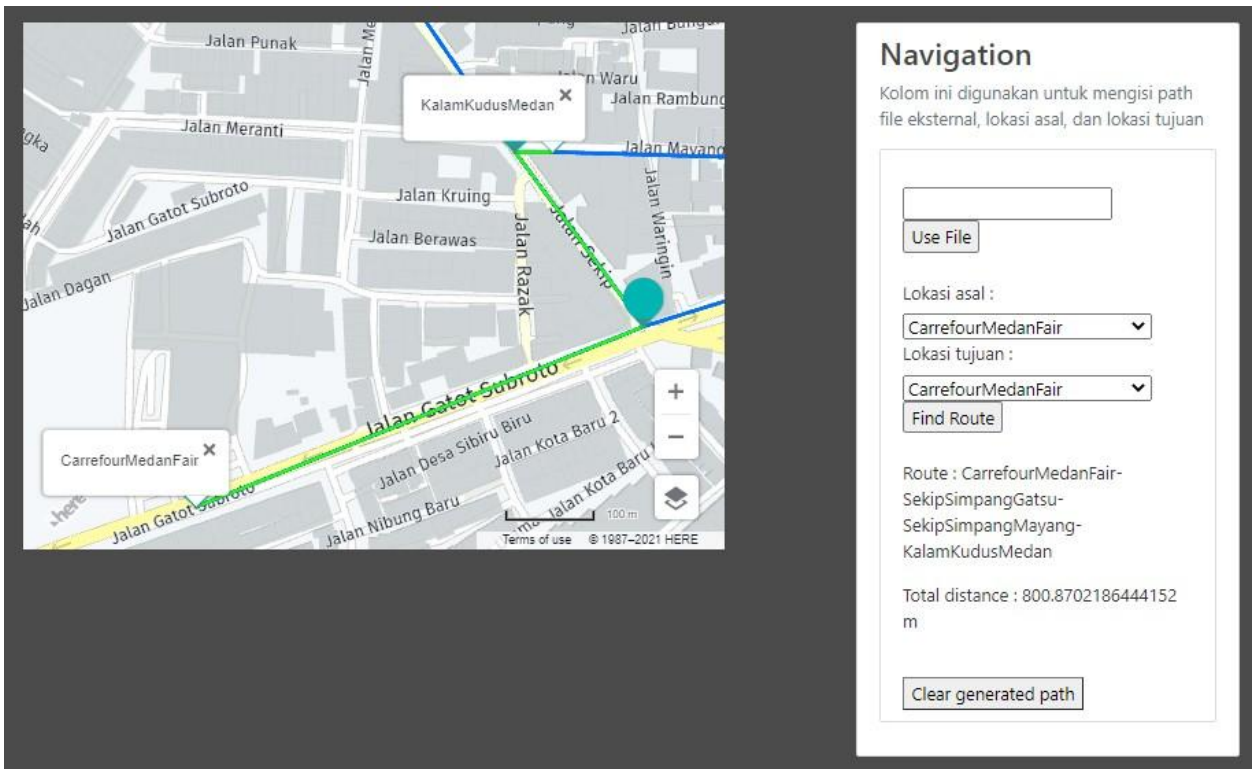
3.2 Testcase 2 : Cafe67 menuju KantorPos



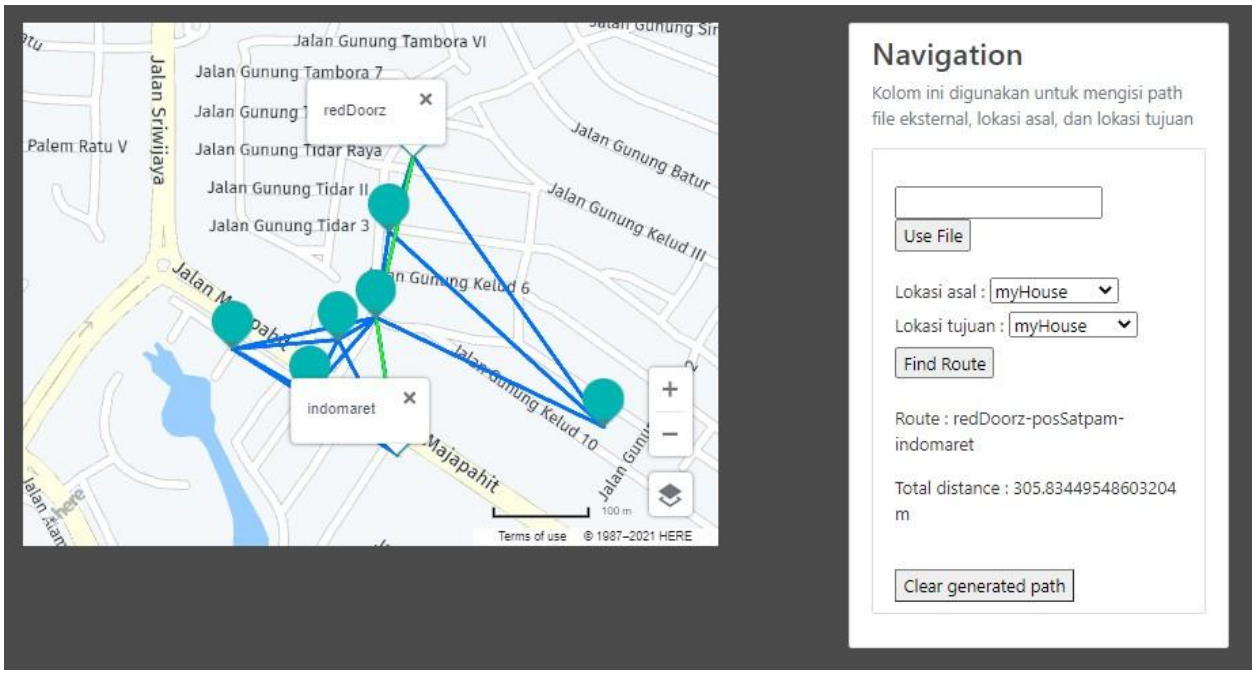
3.3 Testcase 3 : MetroIndahMall menuju TamanS2



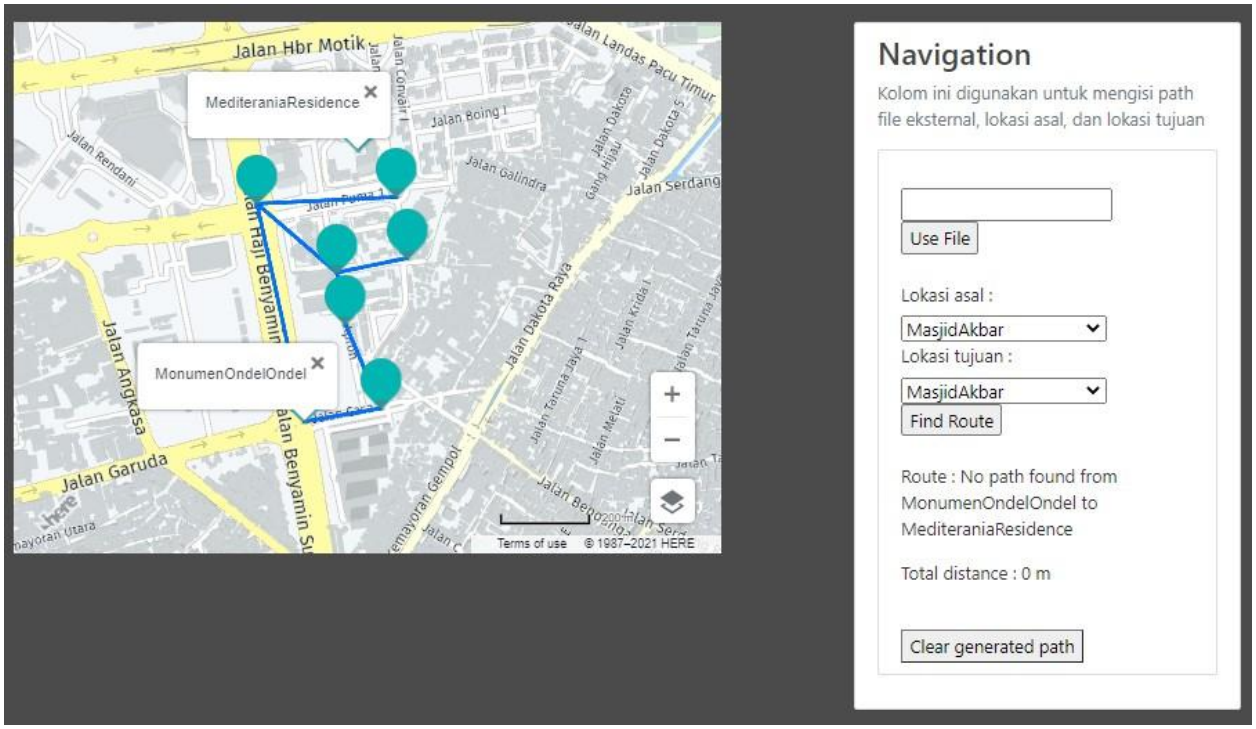
3.4 Testcase 4 : CarrefourMedanFair menuju KalamKudusMedan



3.5 Testcase 5 : redDoorz menuju Indomaret



3.6 Testcase 6 : MonumenOndelOndel menuju MediteraniaResidence



BAB IV ALAMAT KODE SUMBER PROGRAM

Alamat kode sumber program dapat diakses melalui :

https://github.com/BeforeLast/PathOfAStar

BAB V TABEL PENILAIAN

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