**Part 1**

**Abstract Data Type: List ADT**

**ADT handle by: Tan Yi Jing , Tai Lok Khee**

**Tittle: List ADT Specification**

**Description**: The main objective of this ADT class is for store generic type object into a list. This class is used to perform add, remove, clear and retrieve the object inside the list. This class able to check the specific object whether is contain inside the list or not.

add (T newEntry)

Description: Add new object to this list.

Pre-condition: Check the list is empty or consist any object inside the list.

Post-condition: number of entries will be increment.

Return: Boolean

remove (int givenPosition)

Description: Using the index receive from argument to remove the object from the list.

Pre-condition: Checking the index is between greater than equal to one and less than equal to number of entries.

Post-condition: number of entries will be decrement.

Return: Generic type object

getEntry (int givenPosition)

Description: Using the index receive from argument to retrieve the object from the list.

Pre-condition: : Checking the index is between greater than equal to one and less than equal to number of entries.

Return: Generic type object

contains (T anEntry)

Description: Check the object from argument it’s contain in the list.

Return: Boolean

getNumberOfEntries ()

Description: Get the total of number entries of list

Return: int of numberOfEntries

isEmpty()

Description: Check the list is empty or not.

Return : Boolean

getFirstNode() ()

Description : get the first object from the list.

Return Node

getLastNode ()

Descripntion: get the last object from the list.

Return Node

**Abstract Data Type: Stack ADT**

**ADT handle by: Chan Mun Fong , Choong Mun Sin**

**Tittle: Stack ADT Specification**

**Description**: The main objective of this ADT class is for store generic type object into a list and reverse the list by pop every object in the list. This class is used to perform push, pop, clear and peek the object inside the list.

push (T newEntry)

Description: To push new object to top of stack list.

Post-condition: current size will be increment.

No return.

pop ()

Description: To pop the top object from the stack list.

Pre-condition: Check the stack list is empty or not.

Post-condition: current size will be decrement.

Return Generic type object.

peek ()

Description: To peek the top object from the stack list.

Pre-condition: Check the stack list is empty or not.

Return Generic type object.

isEmpty ()

Pre-condition: Check the top object from the stack is empty or not.

Return: Boolean

getCurrsize ()

Return: int of current size

**Part 2**

**Linked list**

Linked list handle by: Tan Yi Jing , Tai Lok Khee

Description: A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration.

1

2

3

4

5

6 package domain;

7

8 /\*\*

9 \*

10 \* **@author** User

11 \*/

12 public class **linkedlist**<T> implements linkedlistInterface<T> {

13

14 private Node firstNode; // reference to first node

15 private int numberOfEntries; // number of entries in list

16 private Node lastNode;

17

18

19

20 public **linkedlist**() {

21 clear();

22 }

23

24 @Override

25 public final void **clear**() {

26 firstNode = null;

27 lastNode = null;

28 numberOfEntries = 0;

29 }

30

31 @Override

32 public boolean **add**(T newEntry) {

33 Node newNode = new Node(newEntry); // create the new node

34

35 if (isEmpty()) // if empty list

36 {

37 firstNode = newNode;

38 } else {

39 lastNode.next= newNode;

40 newNode.previous = lastNode;

41 }

42 lastNode=newNode;

43 numberOfEntries++;

44 return true;

45 }

46

47 @Override

48 public boolean **add**(int newPosition, T newEntry) { // OutOfMemoryError possible

49 boolean isSuccessful = true;

50

51 if ((newPosition >= 1) && (newPosition <= numberOfEntries + 1)) {

52 Node newNode = new Node(newEntry);

53

54 // if (isEmpty() || (newPosition == 1)) { // case 1: add to beginning of list

55 // newNode.next = firstNode;

56 // firstNode = newNode;

57 // } else { // case 2: list is not empty and newPosition > 1

58 // Node nodeBefore = firstNode;

59 // for (int i = 1; i < newPosition - 1; ++i) {

60 // nodeBefore = nodeBefore.next; // advance nodeBefore to its next node

61 // }

62 //

63 // newNode.next = nodeBefore.next; // make new node point to current node at newPosition

64 // nodeBefore.next = newNode; // make the node before point to the new node

65 // }

66

67

68 if(isEmpty()){

69 firstNode = newNode;

70 lastNode=newNode;

71 }else if ( newPosition == 1 ){

72 firstNode.previous = newNode;

73 newNode.next = firstNode;

74 firstNode = newNode;

75 }else if (newPosition == numberOfEntries){

76 lastNode.next= newNode;

77 newNode.previous= lastNode;

78 lastNode = newNode;

79 }

80 else

81 {

82 Node nodeBefore = firstNode;

83 for (int i = 1; i < newPosition - 1; ++i) {

84 nodeBefore = nodeBefore.next; // advance nodeBefore to its next node

85 }

86 newNode.next = nodeBefore.next; // make new node point to current node at newPosition

87 nodeBefore.next.previous= newNode;

88 newNode.previous= nodeBefore;

89 nodeBefore.next = newNode;

90 }

91

92 numberOfEntries++;

93 } else {

94 isSuccessful = false;

95 }

96

97 return isSuccessful;

98 }

99

100

101

102 @Override

103 public T **remove**(int givenPosition) {

104 T result = null; // return value

105

106 if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {

107 if (givenPosition == 1) { // case 1: remove first entry

108 result = firstNode.data; // save entry to be removed

109 firstNode = firstNode.next;

110 }else if (givenPosition == numberOfEntries){

111 result = lastNode.data;

112 lastNode = lastNode.previous;

113 lastNode.next=null;

114 }

115 else { // case 2: givenPosition > 1

116 Node nodeBefore = firstNode;

117 for (int i = 1; i < givenPosition - 1; ++i) {

118 nodeBefore = nodeBefore.next; // advance nodeBefore to its next node

119 }

120 result = nodeBefore.next.data; // save entry to be removed

121 nodeBefore.next = nodeBefore.next.next;

122 nodeBefore.next.next.previous = nodeBefore;

123 // make node before point to node after the

124 } // one to be deleted (to disconnect node from chain)

125

126 numberOfEntries--;

127 }

128

129 return result; // return removed entry, or

130 // null if operation fails

131 }

132

133 @Override

134 public boolean **replace**(int givenPosition, T newEntry) {

135 boolean isSuccessful = true;

136

137 if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {

138 Node currentNode = firstNode;

139 for (int i = 0; i < givenPosition - 1; ++i) {

140 // System.out.println("Trace| currentNode.data = " + currentNode.data + "\t, i = " + i);

141 currentNode = currentNode.next; // advance currentNode to next node

142 }

143 currentNode.data = newEntry; // currentNode is pointing to the node at givenPosition

144 } else {

145 isSuccessful = false;

146 }

147

148 return isSuccessful;

149 }

150

151 @Override

152 public T **getEntry**(int givenPosition) {

153 T result = null;

154

155 if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {

156 Node currentNode = firstNode;

157 for (int i = 0; i < givenPosition - 1; ++i) {

158 currentNode = currentNode.next; // advance currentNode to next node

159 }

160 result = currentNode.data; // currentNode is pointing to the node at givenPosition

161 }

162

163 return result;

164 }

165

166 @Override

167 public boolean **contains**(T anEntry) {

168 boolean found = false;

169 Node currentNode = firstNode;

170

171 while (!found && (currentNode != null)) {

172 if (anEntry.equals(currentNode.data)) {

173 found = true;

174 } else {

175 currentNode = currentNode.next;

176 }

177 }

178

179 return found;

180 }

181

182 @Override

183 public int **getNumberOfEntries**() {

184 return numberOfEntries;

185 }

186

187 @Override

188 public boolean **isEmpty**() {

189 boolean result;

190

191 result = numberOfEntries == 0;

192

193 return result;

194 }

195

196 @Override

197 public boolean **isFull**() {

198 return false;

199 }

200

201 @Override

202 public String **toString**() {

203 String outputStr = "";

204 Node currentNode = firstNode;

205 while (currentNode != null) {

206 outputStr += currentNode.data + "**\n**";

207 currentNode = currentNode.next;

208 }

209 return outputStr;

210 }

211

212 public String **toReverse**(){

213 String outputStr = "";

214 Node currentNode = lastNode;

215 while (currentNode != null) {

216 outputStr += currentNode.data + "**\n**";

217 currentNode = currentNode.previous;

218 }

219 return outputStr;

220 }

221

222 public Node **getFirstNode**() {

223 return firstNode;

224 }

225

226 public Node **getLastNode**() {

227 return lastNode;

228 }

229

230 public class **Node** {

231

232 private T data;

233 private Node previous;

234 private Node next;

235

236 private **Node**(T data) {

237 this.data = data;

238 this.next = null;

239 this.previous=null;

240 }

241

242 private **Node**(T data, Node next) {

243 this.data = data;

244 this.next = next;

245 }

246

247 public T **getData**() {

248 return data;

249 }

250

251

252 } // end Node

253

254 } // end LList

255

**Stack linked list**

Stack linked list handle by: Chan Mun Fong , Choong Mun Sin

Description:  a [stack](http://www.geeksforgeeks.org/stack-data-structure/) using single linked list concept. All the single [linked list](http://www.geeksforgeeks.org/data-structures/linked-list/) operations perform based on Stack operations LIFO(last in first out). The main advantage of using linked list over an arrays is that it is possible to implements a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocate. so overflow is not possible.

package domain;

7

8 /\*\*

9 \*

10 \* @author User

11 \*/

12 public class LinkedStack<T> implements StackInterface<T>{

13 private Node topNode;

14 private int currsize;

15

16 public LinkedStack(){

17 topNode = null;

18 currsize=0;

19 }

20

21 @Override

22 public void push(T newEntry) {

23 Node newNode = new Node(newEntry,topNode);

24 topNode=newNode;

25 currsize++;

26 }

27

28 @Override

29 public T pop() {

30 T data = peek();

31 if(!isEmpty()){

32 data = topNode.data;

33 topNode = topNode.next;

34 currsize--;

35 }

36

37 return data;

38 }

39

40 @Override

41 public T peek() {

42 if(topNode==null)

43 return null;

44 else

45 return topNode.data;

46 }

47

48 @Override

49 public boolean isEmpty() {

50 return topNode == null;

51 }

52

53 @Override

54 public void clear() {

55 topNode =null; //To change body of generated methods, choose Tools | Templates.

56 }

57

58 @Override

59 public int getCurrsize() {

60 return currsize;

61 }

62

63

64

65

66

67 private class Node{

68 private T data;

69 private Node next;

70

71 public Node(T data) {

72 this.data = data;

73 this.next=null;

74 }

75

76 public Node(T data, Node next) {

77 this.data = data;

78 this.next = next;

79 }

80

81

82 }

83 }

84

**Part 3**

Name: Chan Mun Fong

**A star algorithm (Fastest path)**

The algorithm is a computer algorithm the program used in pathfinding and path traversal. The algorithm plots a walkable path between multiple nodes or points (travel legs) in our “travelling graph”. A star algorithm also has the characteristics of a breath-first search, but the only difference is that A star algorithm compares the weight or distance between each path, hence finding the fastest path. This algorithm is specially design for path finding in terms of the distance, it will not traverse through all the nodes (travel legs) and compare one by one. The main feature of the code includes flexibility since all the distance are not hardcoded in the database hence it provides high maintainability rate.

Source Code:

1 /\*

2 \* To change this license header, choose License Headers in Project Properties.

3 \* To change this template file, choose Tools | Templates

4 \* and open the template in the editor.

5 \*/

6 package domain;

7

8 /\*\*

9 \*

10 \* **@author** User

11 \*/

12 public class **Location** {

13 private double lat;

14 private double lon;

15 private String name;

16 private int code\_num;

17 private char type; // s m l n p k

18 // private char geography ; // c s hb

19 private boolean t\_station;

20 private boolean b\_station;

21 private boolean flight ;

22 private String link;

23

24 private List <Location> adjencies = new List <Location>();

25 private double g\_score;

26 private double h\_score;

27 private double f\_score;

28 private Location parent;

29 private int upper;

30 private int adjency;

31

32 public **Location**() {

33 this.g\_score = 0;

34 this.h\_score = 0;

35 this.f\_score = 0;

36 this.parent = null;

37 code\_num= 0;

38 }

39 // for driver program testing

40 public **Location**(double lat, double lon, String name, char type/\*, char geography\*/) {

41 this();

42 this.lat = lat;

43 this.lon = lon;

44 this.name = name;

45 this.type = type;

46 // this.geography = geography;

47

48 }

49

50 public **Location**( int code\_num,String name) {

51 this.name = name;

52 this.code\_num = code\_num;

53 }

54

55 public **Location**(int code\_num, String name, int upper, String link) {

56 this.name = name;

57 this.code\_num = code\_num;

58 this.link = link;

59 this.upper = upper;

60 }

61

62

63 // real

64 public **Location**(double lat, double lon, String name, int code\_num, char type /\*, char geography\*/, boolean t\_station,boolean b\_station,boolean flight,int upper,String link) {

65 this.lat = lat;

66 this.lon = lon;

67 this.name = name;

68 this.code\_num = code\_num;

69 this.type = type;

70 // this.geography = geography;

71 this.flight = flight;

72 this.t\_station = t\_station;

73 this.b\_station = b\_station;

74 this.upper=upper;

75 this.link=link;

76 }

77 //yj

78 public **Location**(double lat, double lon, String name, int code\_num, char type , boolean t\_station,boolean b\_station,boolean flight,int upper,String link,int adjency) {

79 this.lat = lat;

80 this.lon = lon;

81 this.name = name;

82 this.code\_num = code\_num;

83 this.type = type;

84 this.flight = flight;

85 this.t\_station = t\_station;

86 this.b\_station = b\_station;

87 this.upper=upper;

88 this.link=link;

89 this.adjency=adjency;

90 }

91 public **Location**(Location loca){

92 this.lat = loca.getLat();

93 this.lon = loca.getLon();;

94 this.name = loca.getName();;

95 this.code\_num = loca.getCode\_num();;

96 this.type = loca.getType();;

97 this.t\_station = loca.isT\_station();;

98 this.b\_station = loca.isB\_station();;

99 this.flight = loca.isFlight();;

100 this.link = loca.getLink();;

101 this.adjency = loca.getAdjency();;

102 this.g\_score = loca.getG\_score();;

103 this.h\_score = loca.getH\_score();;

104 this.f\_score = loca.getF\_score();;

105 this.parent = loca.getParent();;

106 this.upper = loca.getUpper();;

107 }

108

109 public void **setLat**(double lat) {

110 this.lat = lat;

111 }

112

113 public void **setLon**(double lon) {

114 this.lon = lon;

115 }

116

117 public void **setName**(String name) {

118 this.name = name;

119 }

120

121 public void **setCode\_num**(int code\_num) {

122 this.code\_num = code\_num;

123 }

124

125

126 public void **setType**(char type) {

127 this.type = type;

128 }

129

130 // public void setGeography(char geography) {

131 // this.geography = geography;

132 // }

133

134 public void **setFlight**(boolean flight) {

135 this.flight = flight;

136 }

137

138

139

140 public void **setT\_station**(boolean t\_station) {

141 this.t\_station = t\_station;

142 }

143

144 public void **setB\_station**(boolean b\_station) {

145 this.b\_station = b\_station;

146 }

147

148 public void **setUpper**(int upper) {

149 this.upper = upper;

150 }

151

152 public void **setAdjencies**(List<Location> adjencies) {

153 this.adjencies = adjencies;

154 }

155

156 public void **setG\_score**(double g\_score) {

157 this.g\_score = g\_score;

158 }

159

160 public void **setH\_score**(double h\_score) {

161 this.h\_score = h\_score;

162 }

163

164 public void **setF\_score**(double f\_score) {

165 this.f\_score = f\_score;

166 }

167

168 public void **setParent**(Location parent) {

169 this.parent = parent;

170 }

171

172

173 public static void ***starsearch***(Location start , Location end){

174 List<Location> explored = new List<Location>();

175 List<Location> openset = new List<Location>();

176 openset.add(start);

177

178 while(openset.size()>0){

179 Location current = openset.getValueOf(1);

180 for(int i = 0 ; i<openset.size();i++){

181 if(openset.getValueOf(i+1).getF\_score() < current.getF\_score() ||openset.getValueOf(i+1).getF\_score() == current.getF\_score() && openset.getValueOf(i+1).getH\_score() < current.getH\_score() )

182 current = openset.getValueOf(i+1);

183 }

184 openset.remove(current);

185 explored.add(current);

186 if(current == end){

187 return;

188 }

189

190 for( int i = 0 ; i < current.getAdjencies().size() ;i++)

191 {

192 Location neighbour = current.getAdjencies().getValueOf(i+1);

193 if(explored.contain(neighbour))

194 continue;

195 double new\_f\_cost=current.getG\_score() + neighbour.distance(current);

196 if(new\_f\_cost<neighbour.getG\_score()|| !openset.contain(neighbour)){

197 neighbour.setG\_score(new\_f\_cost);

198 neighbour.setH\_score(neighbour.distance(end));

199 neighbour.setParent(current);

200

201 if(!openset.contain(neighbour))

202 openset.add(neighbour);

203 }

204

205

206 }

207

208 }

209 }

210

211

212

213 public double **distance**(Location endlo) {

214 if ((lat == endlo.getLat()) && (lon == endlo.getLon())) {

215 return 0;

216 } else {

217

218 double theta = this.lon - endlo.getLon();

219 double dist = Math.*sin*(Math.*toRadians*(lat)) \* Math.*sin*(Math.*toRadians*(endlo.getLat())) + Math.*cos*(Math.*toRadians*(lat)) \* Math.*cos*(Math.*toRadians*(endlo.getLat())) \* Math.*cos*(Math.*toRadians*(theta));

220 dist = Math.*acos*(dist);

221 dist = Math.*toDegrees*(dist);

222 dist = dist \* 60 \* 1.1515;

223 //if (unit == "K") {

224 dist = dist \* 1.609344;

225 // } else if (unit == "N") {

226 // dist = dist \* 0.8684;

227 // }

228

229 return (dist);

230 }

231 }

232 public boolean **equalcode**(int code){

233 return this.code\_num==code;

234 }

235

236 public static double ***adjust***(Location start, Location end , linkedlist<Location> result\_link\_loc) {

237 LinkedStack<Location> stack\_loc = new LinkedStack<>();

238 double cost = 0;

239 System.*out*.println("/");

240 System.*out*.println(start.toString());

241 System.*out*.println(end.toString());

242 System.*out*.println("/");

243 // System.out.println(end.getName());

244 for (Location node = end; node != start; node = node.getParent()) {

245 stack\_loc.push(node);

246 // System.out.println(node.toString());

247 }

248 int size = stack\_loc.getCurrsize();

249 for(int i = 0 ; i < size;i++){

250 // System.out.println(stack\_loc.getCurrsize());

251 result\_link\_loc.add(stack\_loc.pop());

252 }

253 return cost;

254 }

255 public static double ***print***(Location start, Location end) {

256 double cost = 0;

257 System.*out*.println(end.getName());

258 for (Location node = end; node != start; node = node.getParent()) {

259 System.*out*.println(node.getParent().getName());

260 cost += node.distance(node.getParent());

261 }

262 return cost;

263 }

264

265 public double **getLat**() {

266 return lat;

267 }

268

269 public double **getLon**() {

270 return lon;

271 }

272

273 public String **getName**() {

274 return name;

275 }

276

277 public int **getCode\_num**() {

278 return code\_num;

279 }

280

281 public int **getUpper**() {

282 return upper;

283 }

284

285 public boolean **isT\_station**() {

286 return t\_station;

287 }

288

289 public boolean **isB\_station**() {

290 return b\_station;

291 }

292

293 public boolean **isFlight**() {

294 return flight;

295 }

296

297 public int **getAdjency**() {

298 return adjency;

299 }

300

301 public char **getType**() {

302 return type;

303 }

304

305 // public char getGeography() {

306 // return geography;

307 // }

308

309 public List<Location> **getAdjencies**() {

310 return adjencies;

311 }

312

313 public double **getG\_score**() {

314 return g\_score;

315 }

316

317 public double **getH\_score**() {

318 return h\_score;

319 }

320

321 public double **getF\_score**() {

322 return f\_score;

323 }

324

325 public Location **getParent**() {

326 return parent;

327 }

328

329 public String **getLink**() {

330 return link;

331 }

332

333 @Override

334 public String **toString**() {

335 return "Location{" + "lat=" + lat + ", lon=" + lon + ", name=" + name + ", code\_num=" + code\_num + ", type=" + type + /\*", geography=" + geography +\*/ ", flight=" + flight + ", station=" + t\_station + '}' +"" ;

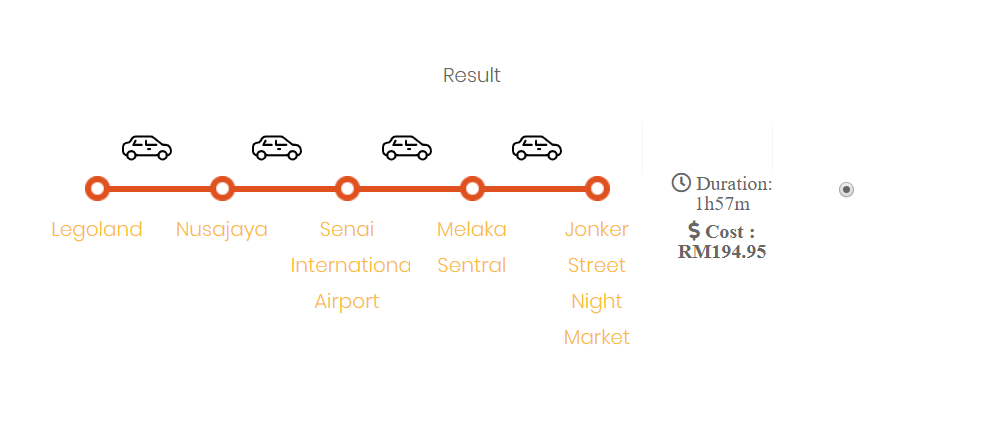
336 }

337

338

339 }

340



The figure above shows that the system will automatically find the shortest and fastest route for the user.

**Domestic and international travel**

Our program allows the user to choose any location as a staring point and any location as a destination. Travel legs between the source and destination can be either local or foreign country. Our program will automatically detect that there is foreign country in the path and will direct the user to the nearest airport. The programme will also show the path to the nearest airport to the destination in order to save time and cost. Hence, the programme is able to accept both local and foreign country or location together in one form.

1 /\*

2 \* To change this license header, choose License Headers in Project Properties.

3 \* To change this template file, choose Tools | Templates

4 \* and open the template in the editor.

5 \*/

6 package DA;

7

8 /\*\*

9 \*

10 \* **@author** User

11 \*/

12 import domain.Location;

13 import domain.List;

14 import domain.linkedlist;

15 import java.sql.Connection;

16 import java.sql.DriverManager;

17 import java.sql.PreparedStatement;

18 import java.sql.ResultSet;

19 import java.sql.SQLException;

20

21 public class **LocationDA** {

22 private String host = "jdbc:derby://localhost:1527/Location";

23 private String user = "nbuser";

24 private String password = "nbuser";

25 private String tableName = "location";

26 private String tableName2 = "ADJENCENT";

27 private Connection conn;

28 private PreparedStatement stmt;

29 private String sqlQueryStr = "SELECT \* from " + tableName;

30 private String sqlQueryStrADJ = "SELECT \* from " + tableName2;

31

32 private String sqlsltsomeType = "SELECT \* from " + tableName + " where type = ? ";

33 // private String sqlsltsomeGeo = "SELECT \* from " + tableName + " where geography = ? ";

34 private String sqlsltsomeparent = "SELECT \* from " + tableName + " where state\_id = ? ";

35 private String sqlsltonename = "SELECT \* from " + tableName + " where name = ? ";

36 private String sqlsltgetairport = "select \* from location where state\_id in ( select state\_id from state where country\_id in ( select country\_id from state where state\_id = ( select state\_id from location where name = ? ) ) ) and flight=true ";

37

38 private String sqlgetalllocationfromcontinent = "SELECT L.L\_CODE , L.LON , L.LAT ,L.**\"**NAME**\"** ,L.**\"**TYPE**\"**, L.T\_STATION, L.B\_STATION, L.FLIGHT,L.STATE\_ID,L.**\"**LINK**\"** FROM continent CT , country C , **\"**STATE**\"** S, location L where CT.CON\_ID=C.CON\_ID and C.COUNTRY\_ID= S.COUNTRY\_ID and S.STATE\_ID = L.STATE\_ID and CT.CON\_ID=?" ;

39 private String sqlgetalllocationfromcountry = "SELECT L.L\_CODE , L.LON , L.LAT ,L.**\"**NAME**\"** ,L.**\"**TYPE**\"**, L.T\_STATION, L.B\_STATION, L.FLIGHT,L.STATE\_ID,L.**\"**LINK**\"** FROM country C , **\"**STATE**\"** S, location L where C.COUNTRY\_ID= S.COUNTRY\_ID and S.STATE\_ID = L.STATE\_ID and C.COUNTRY\_ID=? ";

40 private String slqgetalllocationfromstate = "SELECT L.L\_CODE , L.LON , L.LAT ,L.**\"**NAME**\"** ,L.**\"**TYPE**\"**, L.T\_STATION, L.B\_STATION, L.FLIGHT,L.STATE\_ID,L.**\"**LINK**\"** FROM **\"**STATE**\"** S, location L where S.STATE\_ID = L.STATE\_ID and S.STATE\_ID =?";

41 private String sqlInsertStr = "INSERT INTO " + tableName + " VALUES(?, ?, ?)";

42

43 private String sqlgetonelocation = "SELECT \* from " + tableName + " where l\_code = ? ";

44

45 private String retrieveSpecificData="Select \* from " +tableName + " where l\_code = ? ";

46 private ResultSet rs;

47

48 public **LocationDA**() {

49 getConnection();

50 }

51

52

53

54 public void **getConnection**() {

55 try{

56 conn=DriverManager.*getConnection*(host, user, password);

57 }catch(SQLException ex){

58 ex.getMessage();

59 }

60 }

61

62

63

64 public Location **getCurrentRecord**() {

65 Location location = null;

66 try {

67 location = new Location(Double.*parseDouble*(rs.getString(3)), Double.*parseDouble*(rs.getString(2)), rs.getString(4)

68 , Integer.*parseInt*(rs.getString(1)),rs.getString(5).charAt(0)

69 ,rs.getString(6)=="true"

70 ,rs.getString(7)=="true",rs.getString(8)=="true",Integer.*parseInt*(rs.getString(9)),rs.getString(10));

71 } catch (SQLException ex) {

72 ex.getMessage();

73 }

74 return location;

75 }

76

77

78 public List<Location>**getsomeLocation** (int operation,String str){

79

80 List<Location> location = new List<Location>();

81 try {

82 String sqlstr="";

83 if( operation== 1 ){

84 sqlstr= sqlsltsomeType;

85 }

86 else if (operation ==3)

87 sqlstr= sqlsltsomeparent;

88 else if (operation==4){

89 sqlstr= sqlgetalllocationfromcontinent;

90 }

91 else if (operation ==5)

92 sqlstr = sqlgetalllocationfromcountry;

93 else if (operation ==6)

94 sqlstr=slqgetalllocationfromstate;

95 stmt = conn.prepareStatement(sqlstr);

96 stmt.setString(1, str);

97 rs = stmt.executeQuery();

98

99 while (rs.next()) {

100 location.add(getCurrentRecord());

101 }

102

103 setAdj(location);

104

105

106 } catch (SQLException ex) {

107 ex.getMessage();

108 }

109

110 return location;

111

112

113 }

114 public List<Location> **getAllLocation**() {

115

116 List<Location> location = new List<Location>();

117 try {

118

119

120 stmt = conn.prepareStatement(sqlQueryStr);

121 rs = stmt.executeQuery();

122 while (rs.next()) {

123 location.add(getCurrentRecord());

124 }

125 setAdj(location);

126

127

128

129 } catch (SQLException ex) {

130 ex.getMessage();

131 }

132

133 return location;

134 }

135

136 public void **setAdj**(List<Location> location){

137 try{

138 stmt = conn.prepareStatement(sqlQueryStrADJ);

139 rs = stmt.executeQuery();

140 while (rs.next()) {

141 for(int i = 0 ; i < location.size() ; i ++)

142 {

143 Location tem\_loc = location.getValueOf(i+1);

144 if(tem\_loc.equalcode(rs.getInt(1))){

145 for(int j = 0 ; j < location.size() ; j ++){

146 Location tem\_loc2 = location.getValueOf(j+1);

147 if(tem\_loc2.equalcode(rs.getInt(2))){

148 tem\_loc.getAdjencies().add(tem\_loc2);

149 break;

150 }

151 }

152 break;

153 }

154

155 }

156 }

157 }

158 catch(SQLException ex){

159 ex.getMessage();

160 }

161 }

162

163 public linkedlist<Location> **getAlllinkLocation**() {

164

165 linkedlist<Location> location = new linkedlist<Location>();

166 try {

167

168 stmt = conn.prepareStatement(sqlQueryStr);

169 rs = stmt.executeQuery();

170 while (rs.next()) {

171 location.add(getCurrentRecord());

172 }

173 setAdj(location);

174

175

176

177 } catch (SQLException ex) {

178 ex.getMessage();

179 }

180

181 return location;

182 }

183

184

185 public void **setAdj**(linkedlist<Location> location){

186 try{

187 stmt = conn.prepareStatement(sqlQueryStrADJ);

188 rs = stmt.executeQuery();

189 while (rs.next()) {

190 for(int i = 0 ; i < location.getNumberOfEntries(); i ++)

191 {

192 Location tem\_loc = location.getEntry(i+1);

193 if(tem\_loc.equalcode(rs.getInt(1))){

194 for(int j = 0 ; j < location.getNumberOfEntries(); j ++){

195 Location tem\_loc2 = location.getEntry(j+1);

196 if(tem\_loc2.equalcode(rs.getInt(2))){

197 tem\_loc.getAdjencies().add(tem\_loc2);

198 break;

199 }

200 }

201 break;

202 }

203

204 }

205 }

206 }

207 catch(SQLException ex){

208 ex.getMessage();

209 }

210 }

211

212 public Location **get\_one\_loc**(String a\_loc){

213 Location loc= new Location();

214 try{

215

216 stmt = conn.prepareStatement(sqlsltonename);

217 stmt.setString(1, a\_loc);

218 rs = stmt.executeQuery();

219 rs.next();

220 loc = getCurrentRecord();

221 }

222 catch(SQLException ex){

223 ex.getMessage();

224 }

225 return loc;

226 }

227

228 public Location **get\_one\_loc**(int operation , String a\_loc){

229 Location loc= new Location();

230 try{

231

232 stmt = conn.prepareStatement(sqlgetonelocation);

233 stmt.setString(1, a\_loc);

234 rs = stmt.executeQuery();

235 rs.next();

236 loc = getCurrentRecord();

237 }

238 catch(SQLException ex){

239 ex.getMessage();

240 }

241 return loc;

242 }

243

244 public Location **get\_nearest\_airport**(String a\_loc){

245 List<Location> loc = new List<Location>();

246 Location near = new Location();

247 Location curr = new Location();

248 try{

249 stmt = conn.prepareStatement(sqlsltgetairport);

250 stmt.setString(1, a\_loc);

251 rs = stmt.executeQuery();

252

253 while (rs.next()) {

254 loc.add(getCurrentRecord());

255 }

256

257 curr = get\_one\_loc(a\_loc);

258 near = loc.getValueOf(1);

259

260 for(int i =0 ; i < loc.size(); i++){

261 if(curr.distance(near) >= curr.distance(loc.getValueOf(i+1))){

262 near = loc.getValueOf(i+1);

263 }

264 }

265 }

266 catch(SQLException ex){

267 ex.getMessage();

268 }

269 return near;

270 }

271

272

273 public Location **getSpecificRecord**(int locationcode){

274

275 Location loca=null;

276

277 try {

278 stmt = conn.prepareStatement(retrieveSpecificData);

279 stmt.setString(1, String.*valueOf*(locationcode));

280 rs = stmt.executeQuery();

281 while(rs.next())

282 { loca = new Location(Double.*parseDouble*(rs.getString(3)), Double.*parseDouble*(rs.getString(2)), rs.getString(4)

283 , Integer.*parseInt*(rs.getString(1)),rs.getString(5).charAt(0)

284 ,rs.getString(6)=="true"

285 ,rs.getString(7)=="true",rs.getString(8)=="true",Integer.*parseInt*(rs.getString(9)),rs.getString(10));

286 }

287 } catch (SQLException ex) {

288 ex.getMessage();

289 }

290

291 return loca;

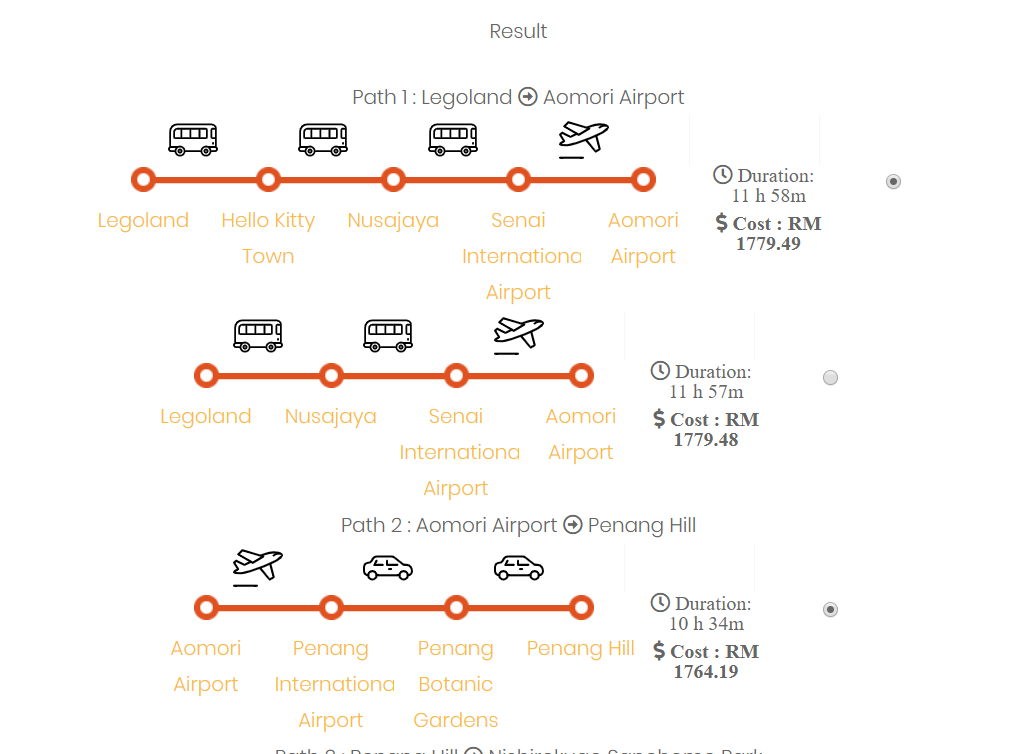
292 }

293

294

295 }

296



The figure above shows that the user is able to travel from local to foreign country and back to local country again in 1 form.

Name: Tan Yi Jing

**Breath first search including recursion (Alternate Path)**

Breath first search algorithm in the program will provide the user alternative path from one specific location to another. While A star algorithm only provides the fastest path, breath first search algorithm main function is to find all possible path from one location to another location. Breath first search time complexity is **O(V + E)**, where V is the number of nodes (location) and E is the number of edges or linking between two locations. This algorithm is also very flexible since we only need to provide the edges between different location.

1 /\*

2 \* To change this license header, choose License Headers in Project Properties.

3 \* To change this template file, choose Tools | Templates

4 \* and open the template in the editor.

5 \*/

6 package domain;

7

8 import DA.adjencyDA;

9 import DA.LocationDA;

10 import DA.categoriesDA;

11 import java.util.ArrayList;

12

13 /\*\*

14 \*

15 \* @author tanyj

16 \*/

17 public class BStar {

18

19

20 // No. of vertices in graph

21 private int v;

22

23 private List<List<Location>> listOfPath;

24 // adjacency list

25 private ArrayList<Integer>[] adjList;

26 private List<Location> newAddressofPathList;

27

28 public BStar(int vertices){

29

30 //initialise vertex count

31 v = vertices;

32 listOfPath = new List<List<Location>>();

33 newAddressofPathList = new List<Location>();

34 // initialise adjacency list

35

36 initAdjList();

37 }

38 public BStar(){

39

40 }

41

42 // initialise adjacency list

43 @SuppressWarnings("unchecked")

44 private void initAdjList()

45 {

46 adjList = new ArrayList[v];

47

48 //ask mf

49 for(int i = 0; i < v; i++)

50 {

51 adjList[i] = new ArrayList<Integer>();

52 }

53 }

54

55 // add edge

56 public void addEdge(int u, int v)

57 {

58

59 adjList[u].add(v);

60 }

61

62 // Prints all paths from source to destination

63 public List<List<Location>> printAllPaths(int s, int d)

64 {

65 adjencyDA adDA = new adjencyDA();

66 LocationDA locaDA = new LocationDA();

67 categoriesDA cda = new categoriesDA();

68 for(int i = 0; i < adDA.getNumofRecords();i++)

69 {

70 addEdge((adDA.getAllRelatedAdjency().getValueOf(i+1).getCode\_num()-5000), (adDA.getAllRelatedAdjency().getValueOf(i+1).getAdjency()-5000));

71 }

72 if (!(cda.checkSameCountry(locaDA.getSpecificRecord(s+5000).getName(), locaDA.getSpecificRecord(d+5000).getName()))){

73 addEdge(locaDA.get\_nearest\_airport(locaDA.getSpecificRecord(s+5000).getName()).getCode\_num()-5000, locaDA.get\_nearest\_airport(locaDA.getSpecificRecord(d+5000).getName()).getCode\_num()-5000);

74 }

75 boolean[] isVisited = new boolean[v];

76 List<Location> pathList = new List<>();

77

78 //add source to path[]

79 pathList.add(locaDA.getSpecificRecord(s+5000));

80

81 //Call recursive

82

83 return printAllPathsUtil(s, d, isVisited, pathList);

84 }

85

86 // A recursive function to print

87 // all paths from 'u' to 'd'.

88 // isVisited[] keeps track of

89 // vertices in current path.

90 // localPathList<> stores actual

91 // vertices in the current path

92 private List<List<Location>> printAllPathsUtil(Integer u, Integer d,

93 boolean[] isVisited,

94 List<Location> localPathList) {

95

96 // Mark the current position

97

98 isVisited[u] = true;

99 LocationDA locaDA=new LocationDA();

100 if (u==d)

101 {

102

103 // if match found then no need to traverse more till depth

104 isVisited[u]= false;

105 List<Location> tempList = new List<Location>();

106 for(int i=0;i<localPathList.size();i++){

107

108 tempList.add(new Location(localPathList.getValueOf(i+1)));

109

110 }

111 listOfPath.add(tempList);

112

113 return listOfPath;

114 }

115

116 // Recur all the vertices

117 // adjacent to current vertex

118 for (Integer i : adjList[u])

119 {

120

121 if (!isVisited[i])

122 {

123 // store current position

124 // in path[]

125 localPathList.add(locaDA.getSpecificRecord(i+5000));

126 printAllPathsUtil(i, d, isVisited, localPathList);

127

128 // remove current position

129 // in path[]

130

131 for(int z =0;z<localPathList.size();z++){

132 if(localPathList.getValueOf(z+1).getName().equals(locaDA.getSpecificRecord(i+5000).getName()))

133 {

134 localPathList.remove(localPathList.getValueOf(z+1));

135 break;

136 }

137

138 }

139

140

141

142

143 }

144 }

145

146 // Mark the current position

147 isVisited[u] = false;

148 return listOfPath;

149

150 }

151

152 public int getV() {

153 return v;

154 }

155

156 public List<List<Location>> getListOfPath() {

157 return listOfPath;

158 }

159

160 public ArrayList<Integer>[] getAdjList() {

161 return adjList;

162 }

163

164 public List<Location> getNewAddressofPathList() {

165 return newAddressofPathList;

166 }

167

168 public void setV(int v) {

169 this.v = v;

170 }

171

172 public void setListOfPath(List<List<Location>> listOfPath) {

173 this.listOfPath = listOfPath;

174 }

175

176 public void setAdjList(ArrayList<Integer>[] adjList) {

177 this.adjList = adjList;

178 }

179

180 public void setNewAddressofPathList(List<Location> newAddressofPathList) {

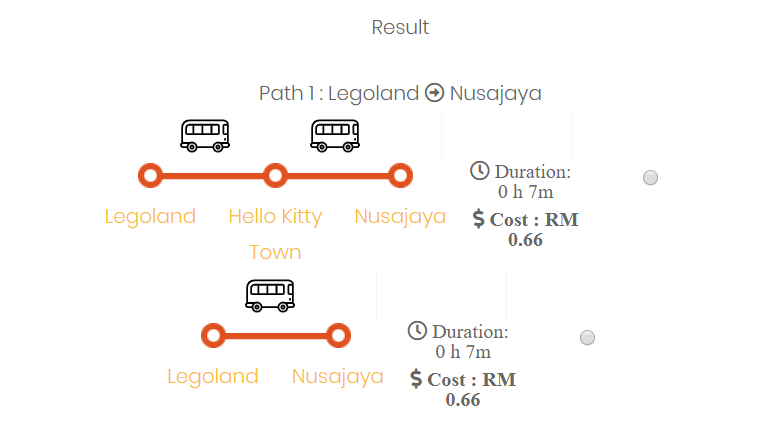
181 this.newAddressofPathList = newAddressofPathList;

182 }

183

184 }

185



The figure above shows that the system returns more than 1 path and show all the alternative path using breath first search.

**Google Maps**

Google Maps API is implemented in the staff side. The staff can enter the source location , destination location and two points and a simple path will be shown to the staff. The changing of the location is real-time meaning that the staff do not need to refresh the page. The staff can keep changing the source or destination location and the pin and the path on the google map will change according to the location that the staff chose real time.

Source Code :

1 **<%--**

2 **Document : AdminSearchTravel**

3 **Created on : Jul 27, 2019, 10:09:09 PM**

4 **Author : tanyj**

5 **--%>**

6

7 **<%@page** contentType**=**"text/html" pageEncoding**=**"UTF-8"**%>**

8 **<%@page** import**=**"domain.List"**%>**

9 **<%@page** import**=**"domain.Location"**%>**

10 **<%@page** import**=**"DA.LocationDA"**%>**

11 <!DOCTYPE html>

12 <html>

13 <head>

14 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

15

16 <style>

17 /\* Set the size of the div element that contains the map \*/

18 #map {

19 height: 400px; /\* The height is 400 pixels \*/

20 width: 100%; /\* The width is the width of the web page \*/

21 }

22 input{

23 margin: 10px;

24 }

25 button{

26 margin: 10px;

27

28 }

29 #map{

30 height: 520px;

31 width: 1500px;

32 }

33

34 body {

35 text-align: center;

36 background: #ebf4fb;

37 min-height: 95vh;

38 margin: 0;

39 padding: 0;

40 border-bottom: 5vh solid #3694d7;

41 font-family: "Myriad Pro","Arial",sans;

42 font-size: 24px;

43 }

44

45 .dropdown-el {

46 margin-top: 20vh;

47 min-width: 12em;

48 position: relative;

49 display: inline-block;

50 margin-right: 1em;

51 min-height: 2em;

52 max-height: 2em;

53 overflow: hidden;

54 top: .5em;

55 cursor: pointer;

56 text-align: left;

57 white-space: nowrap;

58 color: #444;

59 outline: none;

60 border: .06em solid transparent;

61 border-radius: 1em;

62 background-color: #cde4f5;

63 transition: 0.3s all ease-in-out;

64 }

65 .dropdown-el input*:focus* + label {

66 background: #def;

67 }

68 .dropdown-el input {

69 width: 1px;

70 height: 1px;

71 display: inline-block;

72 position: absolute;

73 opacity: 0.01;

74 }

75 .dropdown-el label {

76 border-top: .06em solid #d9d9d9;

77 display: block;

78 height: 2em;

79 line-height: 2em;

80 padding-left: 1em;

81 padding-right: 3em;

82 cursor: pointer;

83 position: relative;

84 transition: 0.3s color ease-in-out;

85 }

86 .dropdown-el label*:nth-child(2)* {

87 margin-top: 2em;

88 border-top: .06em solid #d9d9d9;

89 }

90 .dropdown-el input*:checked* + label {

91 display: block;

92 border-top: none;

93 position: absolute;

94 top: 0;

95 width: 100%;

96 }

97 .dropdown-el input*:checked* + label*:nth-child(2)* {

98 margin-top: 0;

99 position: relative;

100 }

101 .dropdown-el*::after* {

102 content: "";

103 position: absolute;

104 right: 0.8em;

105 top: 0.9em;

106 border: 0.3em solid #3694d7;

107 border-color: #3694d7 transparent transparent transparent;

108 transition: .4s all ease-in-out;

109 }

110 .dropdown-el.expanded {

111 border: 0.06em solid #3694d7;

112 background: #fff;

113 border-radius: .25em;

114 padding: 0;

115 box-shadow: rgba(0, 0, 0, 0.1) 3px 3px 5px 0px;

116 max-height: 15em;

117 }

118 .dropdown-el.expanded label {

119 border-top: .06em solid #d9d9d9;

120 }

121 .dropdown-el.expanded label*:hover* {

122 color: #3694d7;

123 }

124 .dropdown-el.expanded input*:checked* + label {

125 color: #3694d7;

126 }

127 .dropdown-el.expanded*::after* {

128 transform: rotate(-180deg);

129 top: .55em;

130 }

131

132 </style>

133

134 </head>

135 <body>

136 <a href = "AdminPage.jsp" class="btn btn-success" > BACK </a>

137

138

139

140

141

142 <form action="adminSearchTravel">

143 <h1>Search Cheapest/Fastest Route Between Two Location</h1>

144 Starting Location :

145 <input list="Startlocation" type="text" name="startlocation" id="start" required>

146 <datalist id="Startlocation">

147

148 <%

149 List<Location> location = new List<Location>();

150 LocationDA locationDA = new LocationDA();

151 location = locationDA.getAllLocation();

152

153 for (int i = 0; i < location.size(); i++) {%>

154 <option value="<%= location.getValueOf(i + 1).getName()%>">

155

156 <% }%>

157

158 </datalist>

159

160 Destination :

161 <input list="Endlocation" type="text" name="endlocation" id="end" required>

162 <datalist id="Endlocation">

163 <%

164 for (int i = 0; i < location.size(); i++) {%>

165 <option value="<%= location.getValueOf(i + 1).getName()%>" >

166

167 <% }%>

168

169 </datalist>

170 <br><br>

171 <input type="radio" name="searchOption" value="Fastest" id="fastest" checked="true">

172 <label for="fastest">Fastest Route</label>

173 <input type="radio" name="searchOption" value="Cheapest" id="chepest" >

174 <label for="chepest">Cheapest Route</label>

175 <br>

176 <button type="submit" >SUBMIT</button>

177 </form>

178 <!--The div element for the map -->

179 <div id="map"></div>

180

181

182 <script>

183 // Initialize and add the map

184 var sourcelat = 2.7456;

185 var sourcelon = 101.7072;

186 var endlat = null;

187 var endlon = null;

188 function **initMap**() {

189 // The location of Uluru

190

191

192

193 var **uluru** = {lat: sourcelat, lng: sourcelon};

194 var **uluru2** = {lat: endlat, lng: endlon};

195 // The map, centered at Uluru

196 var map = new google.maps.Map(

197 document.getElementById('map'), {zoom: 10, center: uluru});

198 // The marker, positioned at Uluru

199 var marker = new google.maps.Marker({position: uluru, map: map});

200 var marker = new google.maps.Marker({position: uluru2, map: map});

201

202 var flightPlanCoordinates = [

203 {lat: sourcelat, lng: sourcelon},

204 {lat: endlat, lng: endlon}

205

206 ];

207

208 var flightPath = new google.maps.Polyline({

209 path: flightPlanCoordinates,

210 geodesic: true,

211 strokeColor: '#FF0000',

212 strokeOpacity: 1.0,

213 strokeWeight: 2

214 });

215 flightPath.setMap(map);

216 }

217

218 function **displayRadioValue**() {

219 document.getElementById("demo").innerHTML = "";

220 var ele = document.getElementsByName('searchOption');

221

222 for (i = 0; i < ele.length; i++) {

223

224 if (ele[i].type = "radio") {

225

226 if (ele[i].checked)

227 document.getElementById("demo").innerHTML

228 += ele[i].name + " Value: "

229 + ele[i].value + "<br>";

230 }

231 }

232

233 }

234

235

236 function **getlatnlon**() {

237 var source = document.getElementById("start").value;

238 var end = document.getElementById("end").value;

239 <% for (int i = 0; i < location.size(); i++) {%>

240 if (source == '<%=location.getValueOf(i + 1).getName()%>') {

241 sourcelat = <%=location.getValueOf(i + 1).getLat()%>

242 sourcelon = <%=location.getValueOf(i + 1).getLon()%>

243 }

244 <%}%>

245

246 <% for (int i = 0; i < location.size(); i++) {%>

247 if (end == '<%=location.getValueOf(i + 1).getName()%>') {

248 endlat = <%=location.getValueOf(i + 1).getLat()%>

249 endlon = <%=location.getValueOf(i + 1).getLon()%>

250 }

251 <%}%>

252 initMap();

253 }

254

255

256

257

258

259

260

261

262 </script>

263 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

264 <script>

265

266 var value = $("#fastest").val();

267

268 $("#chepest").click(function () {

269 value = $("#chepest").val();

270 });

271

272 $("#fastest").click(function () {

273 value = $("#fastest").val();

274 });

275

276

277

278

279

280

281 $("#start").change(function (e) {

282

283 source = document.getElementById("start").value;

284

285

286 <% for (int i = 0; i < location.size(); i++) {%>

287 if (source == '<%=location.getValueOf(i + 1).getName()%>') {

288 sourcelat = <%=location.getValueOf(i + 1).getLat()%>

289 sourcelon = <%=location.getValueOf(i + 1).getLon()%>

290

291 }

292 <%}%>

293 initMap();

294

295 });

296

297

298

299 $("#end").change(function (e) {

300

301 end = document.getElementById("end").value;

302 <% for (int i = 0; i < location.size(); i++) {%>

303 if (end == '<%=location.getValueOf(i + 1).getName()%>') {

304 endlat = <%=location.getValueOf(i + 1).getLat()%>

305 endlon = <%=location.getValueOf(i + 1).getLon()%>

306 }

307 <%}%>

308 initMap();

309 });

310

311

312 $("#test").click(function (e) {

313 e.preventDefault();

314 $.ajax({

315 url: "adminSearchTravel",

316 // type: "post",

317 **data**: {},

318 cache: false,

319 **success**: function (data) {

320

321 $("#mfmf").html(data).slideDown('slow');

322 }

323 });

324 });

325

326

327 </script>

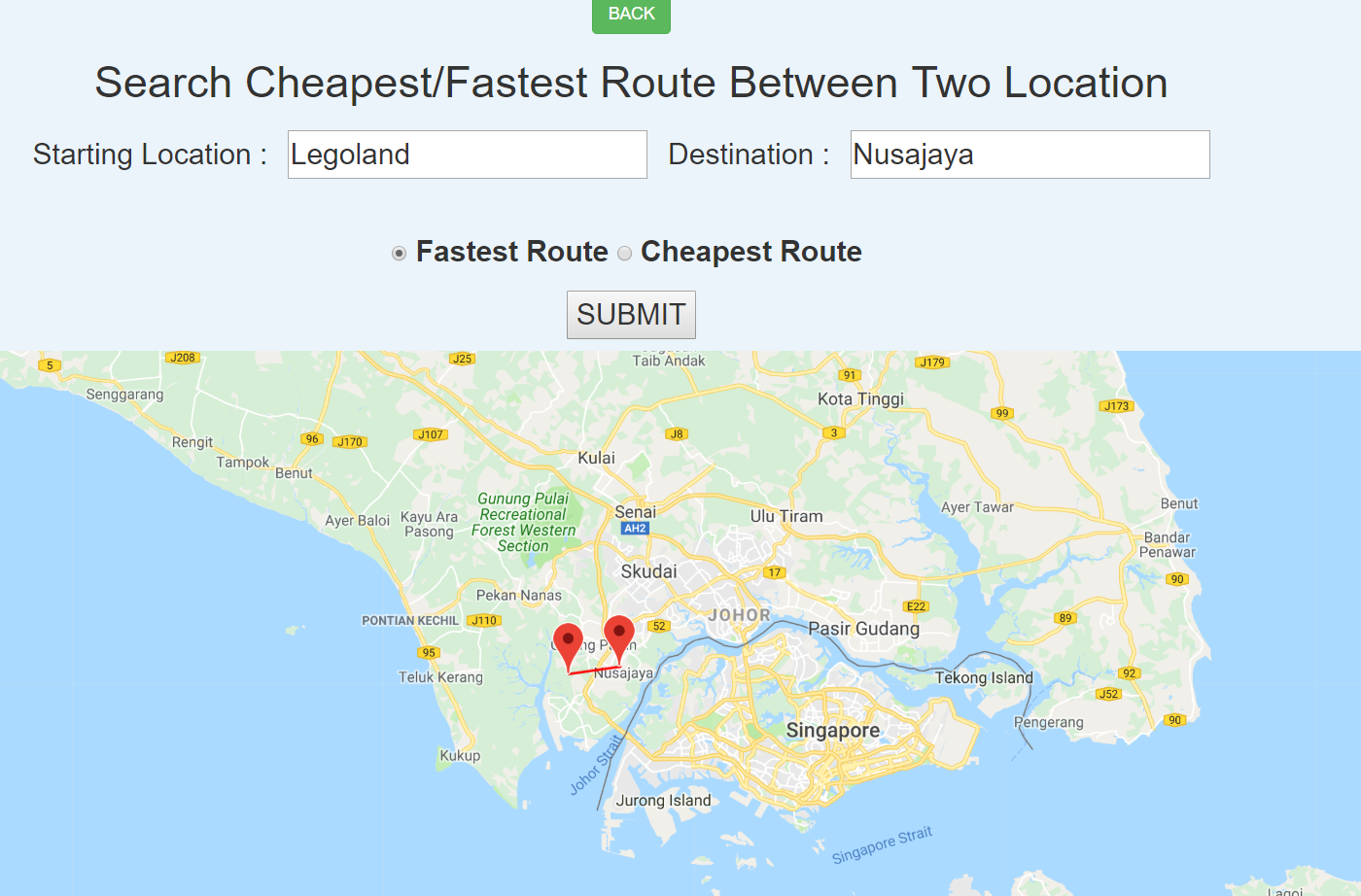
328 <script defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBMrN4VR7U7lR2cxs4VgQ4E8vra5L6A1gk&callback=initMap">

329 </script>

330

331 </body>

332 </html>



The figure above shows the google map functionality.

Name : Tai Lok Khee

**2D List & 3D List**

2D list is created to store all the path return by the breath first search algorithm. Since the breath first search algorithm will return alternative paths from one location to another hence a list of lists of paths is require. The 2D list will contains all of the path from one location to another location. 3D list is created to store all the path in 2D and used to group all path that having same depart and destination location into one list so that can differentiate the path return by the breath first search when the user select more than two location.

2D List Source Code :

7 **<%@page** import**=**"DA.adjencyDA"**%>**

8 **<%@page** import**=**"DA.LocationDA"**%>**

9 **<%@page** import**=**"domain.Location"**%>**

10 **<%@page** import**=**"domain.List"**%>**

11 **<%@page** contentType**=**"text/html" pageEncoding**=**"UTF-8"**%>**

12 <!DOCTYPE html>

13 <html>

14 <head>

15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

16 <title>JSP Page</title>

17 </head>

18 <body>

19

20 <html>

21 <head>

22 <title>Angular & PHP Crud</title>

23 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

24 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

25 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

26 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

27 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

28 <script src="app.js"></script>

29 </head>

30

31 <body **ng-app**="myApp" **ng-controller**="myController">

32 <a href="AdminPage.jsp" class="btn btn-success" > BACK </a>

33

34 <div class="container">

35

36 <hr>

37

38 <h1>All Travel Leg</h1>

39 <span class="clearfix"></span>

40 <hr>

41

42

43 <div class="table-responsive">

44

45 <%

46 if (session.getAttribute("error") != null) {%>

47 <div class="alert alert-danger alert-dismissable" **ng-if**="messageInfo">

48 <a href="#" class="close" data-dismiss="alert" aria-label="close">**&times;**</a>

49 <%= session.getAttribute("error").toString()%>

50 </div>

51 <%}%>

52 <% session.removeAttribute("error"); %>

53 <%

54 if (session.getAttribute("sucess") != null) {%>

55 <div class="alert alert-success alert-dismissable" **ng-if**="messageInfo">

56 <a href="#" class="close" data-dismiss="alert" aria-label="close">**&times;**</a>

57 <%= session.getAttribute("sucess").toString()%>

58 </div>

59 <%}

60 session.removeAttribute("sucess");%>

61

62

63 <table class="table table-striped table-responsive table-hover ">

64 <thead>

65 <tr class="info">

66 <th>Location ID</th>

67 <th>Location Name</th>

68 <th>Current Travel Leg Number</th>

69 <th>View</th>

70 <th>Add</th>

71 <th>Delete</th>

72 </tr>

73 </thead>

74 <tbody>

75 <% LocationDA locaDA = new LocationDA();

76 adjencyDA adDA = new adjencyDA();

77 List<Location> locaList = new List<Location>();

78 locaList = locaDA.getAllLocation();

79 for (int i = 0; i < locaList.size(); i++) {%>

80

81 <tr **ng-repeat**="user in users">

82

83 <td><%= locaList.getValueOf(i + 1).getCode\_num()%> </td>

84 <td><%= locaList.getValueOf(i + 1).getName()%></td>

85 <td><%= adDA.getNumberofTravelLeg(locaList.getValueOf(i + 1).getCode\_num())%></td>

86 <td><button type="button" class="btn btn-success view" data-toggle="modal" data-target="#View<%=locaList.getValueOf(i + 1).getCode\_num()%>" >View</button></td>

87 <td><button type="button" class=" btn btn-info editbtn" data-toggle="modal" data-target="#modaledit" **ng-click**="selectUser(user)" value="<%= locaList.getValueOf(i + 1).getCode\_num()%>" >Edit</button></td>

88

89 <td><button type="button" class="btn btn-danger delete" data-toggle="modal" data-target="#<%=locaList.getValueOf(i + 1).getCode\_num()%>" **ng-click**="selectUser(user)">Delete</button></td>

90

91 </tr>

92 <%}%>

93 </tbody>

94 </table>

95 </div>

96 </div>

97

98

99 <%for (int i = 0; i < locaList.size(); i++) {%>

100 <div class="modal fade" id="View<%= locaList.getValueOf(i + 1).getCode\_num()%>" role="dialog">

101 <div class="modal-dialog">

102 <div class="modal-content">

103 <div class="modal-header">

104 <h4 class="modal-title">View All Travel Leg of <%= locaList.getValueOf(i + 1).getName()%></h4>

105 </div>

106

107 <table class="table table-striped table-responsive table-hover ">

108 <thead>

109 <tr class="info">

110 <th>Location ID</th>

111 <th>Location Name</th>

112 </tr>

113 </thead>

114 <%

115 List<Location> viewAll = adDA.selectDeleteLocation(String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()), String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()));

116 for (int j = 0; j < viewAll.size(); j++) {

117 %>

118 <tr>

119 <td><%= viewAll.getValueOf(j + 1).getCode\_num()%></td>

120 <td><%= viewAll.getValueOf(j + 1).getName()%></td>

121 </tr>

122 <%}%>

123 </table>

124

125

126

127

128 <div class="modal-footer">

129 <button type="submit" class="btn btn-info pull-right" data-dismiss="modal" >Close</button>

130 </div>

131 </div>

132 </div>

133 </div>

134 <%}%>

135

136

137

138

139 <div class="modal fade" id="modaledit" role="dialog">

140 <div class="modal-dialog">

141 <div class="modal-content">

142 <div class="modal-header">

143 <h4 class="modal-title">Add New Travel Leg</h4>

144 <h5>Editing: <div id="sourceLoca"></div></h5>

145 </div>

146 <div class="modal-body">

147 <form class="form-horizontal" action="editServelet">

148 <input id="sourceCode" name="sourceCode" hidden>

149 <div class="form-group">

150 <label class="control-label col-md-2">Location Name</label>

151 <div class="col-md-10">

152 <input list="location" type="text" id="locaName" name="locaName" class=" form-control">

153 <datalist id="location">

154 <%for (int i = 0; i < locaList.size(); i++) {%>

155

156 <option value="<%= locaList.getValueOf(i + 1).getName()%>" >

157 <%}%>

158 </datalist>

159 </div>

160 </div>

161

162 <div class="form-group">

163 <label class="control-label col-md-2">Location Code</label>

164 <div class="col-md-10">

165 <input type="text" class=" form-control" id="locaCode" name="locaCode" readonly="true">

166 </div>

167 </div>

168

169 <div class="form-group">

170 <div class="col-md-2 col-md-offset-2">

171 <button type="submit" class="btn btn-info " name="button" value="update">ADD</button>

172 </div>

173 </div>

174 </form>

175 </div>

176 <div class="modal-footer">

177 <button type="submit" class="btn btn-info pull-right" data-dismiss="modal" >Close</button>

178 </div>

179 </div>

180 </div>

181 </div>

182

183

184

185 <%for (int i = 0; i < locaList.size(); i++) {%>

186 <form action="editServelet">

187 <div class="modal fade" id="<%= locaList.getValueOf(i + 1).getCode\_num()%>" role="dialog">

188 <div class="modal-dialog">

189 <div class="modal-content">

190 <div class="modal-header">

191 <h4 class="modal-title">Deleting Travel Leg</h4>

192 </div>

193 <div class="modal-body">

194 <input list="<%=locaList.getValueOf(i + 1).getName()%>" type="text" class=" form-control" name="delDest">

195 <datalist id="<%=locaList.getValueOf(i + 1).getName()%>">

196 <%

197 List<List<Location>> delLoca = new List<List<Location>>();

198 delLoca.add(adDA.selectDeleteLocation(String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()), String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num())));

199 for (int j = 0; j < delLoca.getValueOf(1).size(); j++) {

200 %>

201

202 <option value="<%= delLoca.getValueOf(1).getValueOf(j + 1).getName()%>">

203 <%}%>

204 </datalist>

205

206 <input value="<%= locaList.getValueOf(i + 1).getCode\_num()%>" name="delSource">

207

208 </div>

209 <div class="modal-footer">

210 <button type="submit" class="btn btn-info " >Yes</button>

211 <button type="button" class="btn btn-info " data-dismiss="modal" >No</button>

212

213 </div>

214 </div>

215 </div>

216 </div>

217 </form>

218 <%}%>

219 </body>

220 </html>

221 </body>

222 </html>

223 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

224 <script>

225

226

227

228

229 $("#locaName").change(function () {

230 var location = document.getElementById("locaName").value;

231

232 <% for (int i = 0; i < locaList.size(); i++) {%>

233 if (location == '<%= locaList.getValueOf(i + 1).getName()%>') {

234 $("#locaCode").val("<%= locaList.getValueOf(i + 1).getCode\_num()%>");

235 }

236 <%}%>

237 }

238 );

239

240 $(".editbtn").click(function (e) {

241 sourceCode = $(this).val();

242 <% for (int i = 0; i < locaList.size(); i++) {%>

243 if (sourceCode == '<%= locaList.getValueOf(i + 1).getCode\_num()%>') {

244 $("#sourceLoca").html("Name : <%= locaList.getValueOf(i + 1).getName()%> <br/> Location Code : <%= locaList.getValueOf(i + 1).getCode\_num()%>");

245 $("#sourceCode").val("<%= locaList.getValueOf(i + 1).getCode\_num()%>");

246 }

247 <%}%>

248

249

250 });

251

252

253

254

255

256 </script>

257

3D List Source Code:

6 package Servlet;

7

8 import DA.LocationDA;

9 import DA.adjencyDA;

10 import DA.categoriesDA;

11 import domain.BStar;

12 import domain.List;

13 import domain.Location;

14 import domain.linkedlist;

15 import java.io.IOException;

16 import java.io.PrintWriter;

17 import javax.servlet.ServletException;

18 import javax.servlet.http.HttpServlet;

19 import javax.servlet.http.HttpServletRequest;

20 import javax.servlet.http.HttpServletResponse;

21 import javax.servlet.http.HttpSession;

22

23 /\*\*

24 \*

25 \* **@author** User

26 \*/

27 public class **userLocProc** extends HttpServlet {

28

29 protected void **processRequest**(HttpServletRequest request, HttpServletResponse response)

30 throws ServletException, IOException {

31 HttpSession s = request.getSession();

32

33 response.setContentType("text/html;charset=UTF-8");

34 String submit = request.getParameter("submit");

35 String time\_Astar = request.getParameter("time");

36 String cost\_Bstart = request.getParameter("cost");

37

38 String src = request.getParameter("country\_id");

39 String reload = request.getParameter("reload");

40

41 LocationDA lda = new LocationDA();

42 categoriesDA cda = new categoriesDA();

43 List<Location> loc = new List<Location>();

44 linkedlist<Location> all\_link\_loc = new linkedlist<Location>();

45 linkedlist<Location> linkloc = new linkedlist<Location>();

46 try (PrintWriter out = response.getWriter()) {

47

48 /\* TODO output your page here. You may use following sample code. \*/

49 if (submit != null) {

50

51 String start = request.getParameter("start\_in");

52 String str[] = request.getParameterValues("mytext[]");

53 String end = request.getParameter("end\_in");

54 if (/\*!time\_Astar.equals("")\*/ request.getParameter("lastchance").equals("2") ) {

55 if (str != null) {

56 for (int i = 0; i < str.length + 1; i++) {

57 if (i == 0) {

58 linkloc.add(lda.get\_one\_loc(start));

59 if (!cda.checkSameCountry(start, str[i])) {

60 linkloc.add(lda.get\_nearest\_airport(start));

61 linkloc.add(lda.get\_nearest\_airport(str[i]));

62 }

63 linkloc.add(lda.get\_one\_loc(str[i]));

64

65 } else if (i == str.length) {

66 if (!(linkloc.getLastNode().getData().getName().equals(lda.get\_one\_loc(str[i - 1]).getName()))) {

67 linkloc.add(lda.get\_one\_loc(str[i - 1]));

68 }

69 if (!cda.checkSameCountry(str[i - 1], end)) {

70 linkloc.add(lda.get\_nearest\_airport(str[i - 1]));

71 linkloc.add(lda.get\_nearest\_airport(end));

72 }

73 linkloc.add(lda.get\_one\_loc(end));

74 } else {

75 if (!(linkloc.getLastNode().getData().getName().equals(lda.get\_one\_loc(str[i - 1]).getName()))) {

76 linkloc.add(lda.get\_one\_loc(str[i]));

77 }

78 if (!cda.checkSameCountry(str[i - 1], str[i])) {

79 linkloc.add(lda.get\_nearest\_airport(str[i - 1]));

80 linkloc.add(lda.get\_nearest\_airport(str[i]));

81 }

82 linkloc.add(lda.get\_one\_loc(str[i]));

83 }

84 }

85 } else {

86 linkloc.add(lda.get\_one\_loc(start));

87 if (!cda.checkSameCountry(start, end)) {

88 linkloc.add(lda.get\_nearest\_airport(start));

89 linkloc.add(lda.get\_nearest\_airport(end));

90 }

91 linkloc.add(lda.get\_one\_loc(end));

92 }

93

94 all\_link\_loc = lda.getAlllinkLocation();

95 System.*out*.println(linkloc.toString());

96 System.*out*.println("next");

97

98 for (int i = 0; i < linkloc.getNumberOfEntries(); i++) {

99 for (int j = 0; j < all\_link\_loc.getNumberOfEntries(); j++) {

100 if (linkloc.getEntry(1).getName().equals(all\_link\_loc.getEntry(j + 1).getName())) {

101 linkloc.add(all\_link\_loc.getEntry(j + 1));

102 linkloc.remove(1);

103 break;

104 }

105 }

106 }

107

108 System.*out*.println(linkloc.toString());

109 linkedlist<Location> result\_loc = new linkedlist<Location>();

110

111 result\_loc.add(linkloc.getEntry(1));

112 for (int i = 0; i < linkloc.getNumberOfEntries() - 1; i++) {

113 if (cda.checkSameCountry(linkloc.getEntry(i + 1).getName(), linkloc.getEntry(i + 2).getName())) {

114 System.*out*.println(cda.checkSameCountry(linkloc.getEntry(i + 1).getName(), linkloc.getEntry(i + 2).getName()));

115 Location.*starsearch*(linkloc.getEntry(i + 1), linkloc.getEntry(i + 2));

116 Location.*adjust*(linkloc.getEntry(i + 1), linkloc.getEntry(i + 2), result\_loc);

117 } else {

118 System.*out*.println(cda.checkSameCountry(linkloc.getEntry(i + 1).getName(), linkloc.getEntry(i + 2).getName()));

119 result\_loc.add(linkloc.getEntry(i + 2));

120 }

121

122 }

123 out.println(result\_loc.toString());

124 System.*out*.println(result\_loc.toString());

125 System.*out*.println("end");

126 HttpSession session = request.getSession();

127 if (!time\_Astar.equals("")) {

128 session.setAttribute("astartime", Double.*parseDouble*(time\_Astar));

129 } else {

130 session.setAttribute("astartime", 9999.99);

131 }

132

133 session.setAttribute("aStar\_Path", result\_loc);

134 response.sendRedirect("userPathProc");

135 } else if (/\*!cost\_Bstart.equals("") \*/ request.getParameter("lastchance").equals("1")) {

136 List<List<List<Location>>> loc2D = new List<List<List<Location>>>();

137 loc = new List<Location>();

138 loc.add(lda.get\_one\_loc(start));

139 if (str != null) {

140 for (int i = 0; i < str.length; i++) {

141 loc.add(lda.get\_one\_loc(str[i]));

142 }

143 }

144 loc.add(lda.get\_one\_loc(end));

145 for (int i = 0; i < loc.size() - 1; i++) {

146 BStar b = new BStar(new adjencyDA().getNumofRecords());

147 b.printAllPaths(loc.getValueOf(i + 1).getCode\_num() - 5000, loc.getValueOf(i + 2).getCode\_num() - 5000);

148 loc2D.add(b.getListOfPath());

149 }

150

151 HttpSession session = request.getSession();

152 session.setAttribute("bStar\_Path", loc2D);

153 if (!cost\_Bstart.equals("")) {

154 session.setAttribute("cost", Double.*parseDouble*(cost\_Bstart));

155 } else {

156 session.setAttribute("cost", 9999.99);

157 }

158

159 response.sendRedirect("userPathProc");

160

161 }

162

163 } else if (reload != null) {

164 loc = cda.getAllcont();

165 for\_printing\_sub\_place(loc, out);

166

167 //System.out.println(6);

168 } else if (src.charAt(0) == '1') {

169 loc = cda.getSomecountry(src);

170 //loc = cda.getAllcont();

171 for\_printing\_sub\_place(loc, out);

172 // System.out.println(2);

173

174 } else if (src.charAt(0) == '2') {

175

176 loc = cda.getSomestate(src);

177 for\_printing\_sub\_place(loc, out);

178

179 } else if (src.charAt(0) == '3') {

180 System.*out*.println(src);

181 loc = lda.getsomeLocation(3, src);

182

183 for (int i = 0; i < loc.size(); i++) {

184 out.println("<div class=**\"**col-md-3 col-sm-6**\"**>**\n**"

185 + " <div class=**\"**product-grid6**\"**>**\n**"

186 + " <div class=**\"**product-image6**\"**>**\n**"

187 + " <a >**\n**"

188 + " <img class=**\"**pic-1**\"** src=**\"**" + loc.getValueOf(i + 1).getLink() + "**\"** >**\n**"

189 + " </a>**\n**"

190 + " **\n**"

191 + " </div>**\n**"

192 + " <div class=**\"**product-content**\"**>**\n**"

193 + " <h3 class=**\"**title**\"**><a > " + loc.getValueOf(i + 1).getName() + "</a></h3>**\n**"

194 + " **\n**"

195 + " </div>**\n**"

196 + " **\n**"

197 + " <ul class=**\"**social**\"**>**\n**"

198 + " <li><button class=**\"**add\_start btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getName() + "**\"** > Start</button></li>**\n**"

199 + " <li><button class=**\"**add\_middle btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getName() + "**\"** > add</button></li>**\n**"

200 + " <li><button class=**\"**add\_end btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getName() + "**\"** > End</button></li>**\n**"

201 + "**\n**"

202 + " </ul>**\n**"

203 + " </div>**\n**"

204 + " </div>");

205 }

206

207 out.println("<script class=**\"**loc\_remove**\"**>**\n**"

208 + "$(document).ready(function() {**\n**"

209 + " var max\_fields = 10; //maximum input boxes allowed**\n**"

210 + " var wrapper = $(**\"**.input\_fields\_wrap**\"**); //Fields wrapper**\n**"

211 + " var add\_button = $(**\"**.add\_middle**\"**); //Add button ID**\n**"

212 + " **\n**"

213 + " var x = 0; //initlal text box count**\n**"

214 + " $(add\_button).click(function(e){ //on add input button click**\n**"

215 + " e.preventDefault();**\n**"

216 + " **\n**"

217 + " if(x < max\_fields){ //max input box allowed**\n**"

218 + " x++; //text box increment**\n**"

219 + " if(x==1)**\n**"

220 + " $(wrapper).append('<label for=**\"**#**\"**>Travel Leg</label>'); **\n**"

221 + " $(wrapper).append('<div><input type=**\"**text**\"** name=**\"**mytext[]**\"** list=**\"**all\_location**\"** value=**\"**'+ $(this).val() +'**\"** /><a href=**\"**#**\"** class=**\"**remove\_field**\"**>Remove</a></div>'); //add input box**\n**"

222 + " } **\n**"

223 + " });**\n**"

224 + " **\n**"

225 + " $(wrapper).on(**\"**click**\"**,**\"**.remove\_field**\"**, function(e){ //user click on remove text**\n**"

226 + " e.preventDefault(); $(this).parent('div').remove(); **\n**"

227 + " x--;**\n**"

228 + " if(x==0){**\n**"

229 + " $(wrapper).children('label').remove(); **\n**"

230 + " }**\n**"

231 + " });**\n**"

232 + " **\n**"

233 + " $(**\"**.add\_start**\"**).click(function(){**\n**"

234 + " var text = $(this).val();**\n**"

235 + " $(**\"**.start\_in**\"**).val(text) ;**\n**"

236 + " })**\n**"

237 + " **\n**"

238 + " $(**\"**.add\_end**\"**).click(function(){**\n**"

239 + " var text = $(this).val();**\n**"

240 + " $(**\"**.end\_in**\"**).val(text) ;**\n**"

241 + " }) **\n**"

242 + " $(**\"**.loc\_remove**\"**).remove(); **\n**"

243 + "});**\n**"

244 + "**\n**"

245 + "</script>");

246

247 }

248

249 }

250

251 }

252

253 public void **for\_printing\_sub\_place**(List<Location> loc, PrintWriter out) {

254

255 for (int i = 0; i < loc.size(); i++) {

256 out.println("<div class=**\"**col-md-3 col-sm-6**\"**>**\n**"

257 + " <div class=**\"**product-grid6**\"**>**\n**"

258 + " <div class=**\"**product-image6**\"**>**\n**"

259 + " <a >**\n**"

260 + " <img class=**\"**pic-1**\"** alt=**\"\"** src=**\"** " + loc.getValueOf(i + 1).getLink() + " **\"** >**\n**"

261 + " </a>**\n**"

262 + " **\n**"

263 + " </div>**\n**"

264 + " <div class=**\"**product-content**\"**>**\n**"

265 + " <h3 class=**\"**title**\"**><a > " + loc.getValueOf(i + 1).getName() + "</a></h3>**\n**"

266 + " **\n**"

267 + " </div>**\n**"

268 + " **\n**"

269 + " <ul class=**\"**social**\"**>**\n**"

270 + " <li><button class=**\"**country\_list btn btn-primary**\"** id=**\"**country\_list**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > next</button></li>**\n**"

271 + " **\n**"

272 + " </ul>**\n**"

273 + " </div>**\n**"

274 + " </div>");

275

276 }

277

278 out.println("<script class=**\"**remove**\"**>$(document).ready(function() {**\n**"

279 + " **\n**"

280 + " $(**\"**.country\_list**\"**).click(function (e) {**\n**"

281 + " var x = $(this).val();**\n**"

282 + " **\n**"

283 + " e.preventDefault();**\n**"

284 + " $.ajax({**\n**"

285 + " url: **\"**userLocProc**\"**,**\n**"

286 + " // type: **\"**post**\"**,**\n**"

287 + " data: {country\_id: x},**\n**"

288 + " cache: false,**\n**"

289 + " success: function(data) {**\n**"

290 + " $(**\"**.row\_my**\"**).html(data).slideDown('slow');**\n**"

291 + " }**\n**"

292 + " });**\n**"

293 + " **\n**"

294 + "});**\n**"

295 + "$(**\"**.remove**\"**).remove();**\n**"

296 + "});**\n**"

297 + "</script>**\n**"

298 + "");

299

300 }

301

302 @Override

303 protected void **doGet**(HttpServletRequest request, HttpServletResponse response)

304 throws ServletException, IOException {

305 processRequest(request, response);

306 }

307

308 @Override

309 protected void **doPost**(HttpServletRequest request, HttpServletResponse response)

310 throws ServletException, IOException {

311 processRequest(request, response);

312 }

313

314 @Override

315 public String **getServletInfo**() {

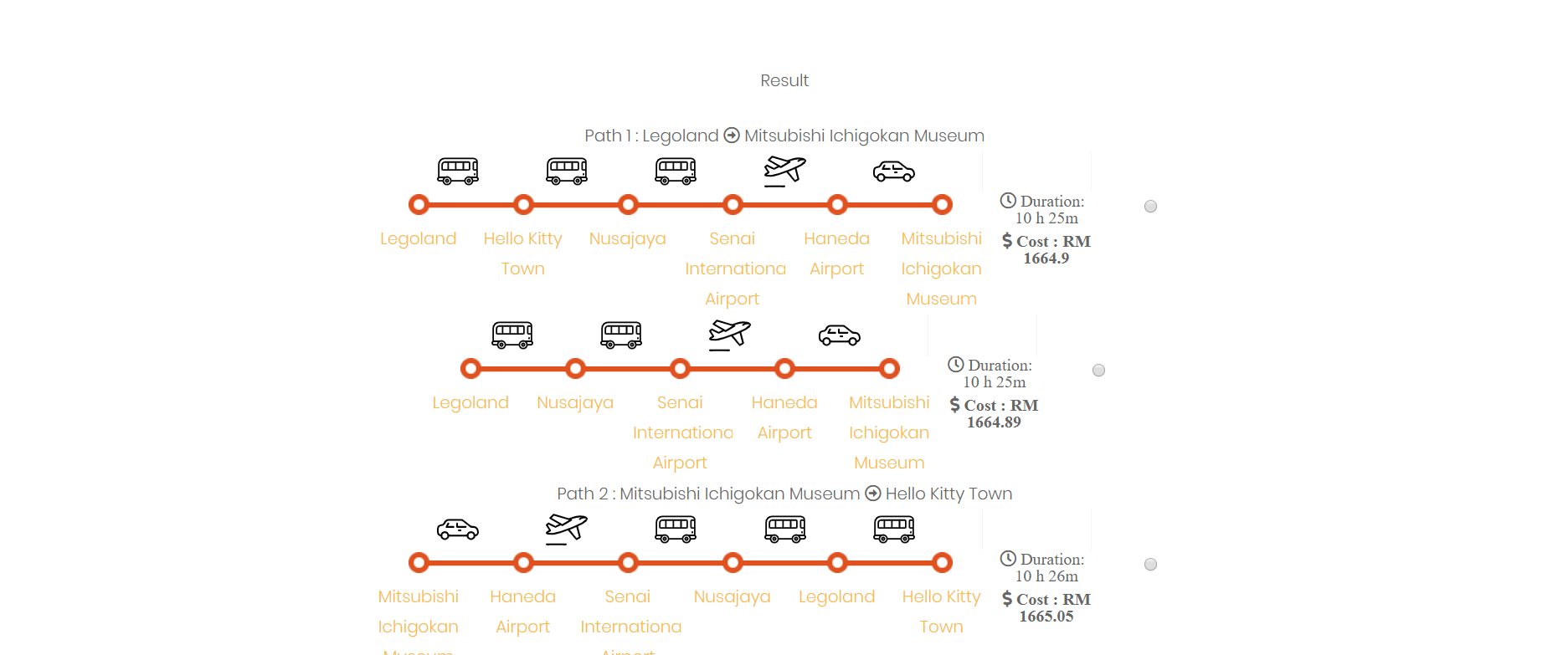
316 return "Short description";

317 }// </editor-fold>

318

319 }

320



The figure above shows that the system return more than more than 1 path of travelling option and path using 2D and 3D List.

**Travel Leg Details**

All the data will be retrieved from the database and store in the ADT. All the processing process such as CRUD regarding on travel leg. The CRUD process will retrieve from the database and store it in the list ADT and use the data in the list ADT to process instead of directly retrieving from the database to do processing.

Source Code :

1 **<%--**

2 **Document : EditTravelLeg**

3 **Created on : Aug 3, 2019, 9:51:50 PM**

4 **Author : tanyj**

5 **--%>**

6

7 **<%@page** import**=**"DA.adjencyDA"**%>**

8 **<%@page** import**=**"DA.LocationDA"**%>**

9 **<%@page** import**=**"domain.Location"**%>**

10 **<%@page** import**=**"domain.List"**%>**

11 **<%@page** contentType**=**"text/html" pageEncoding**=**"UTF-8"**%>**

12 <!DOCTYPE html>

13 <html>

14 <head>

15 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

16 <title>JSP Page</title>

17 </head>

18 <body>

19

20 <html>

21 <head>

22 <title>Angular & PHP Crud</title>

23 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

24 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

25 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

26 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

27 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

28 <script src="app.js"></script>

29 </head>

30

31 <body **ng-app**="myApp" **ng-controller**="myController">

32 <a href="AdminPage.jsp" class="btn btn-success" > BACK </a>

33

34 <div class="container">

35

36 <hr>

37

38 <h1>All Travel Leg</h1>

39 <span class="clearfix"></span>

40 <hr>

41

42

43 <div class="table-responsive">

44

45 <%

46 if (session.getAttribute("error") != null) {%>

47 <div class="alert alert-danger alert-dismissable" **ng-if**="messageInfo">

48 <a href="#" class="close" data-dismiss="alert" aria-label="close">**&times;**</a>

49 <%= session.getAttribute("error").toString()%>

50 </div>

51 <%}%>

52 <% session.removeAttribute("error"); %>

53 <%

54 if (session.getAttribute("sucess") != null) {%>

55 <div class="alert alert-success alert-dismissable" **ng-if**="messageInfo">

56 <a href="#" class="close" data-dismiss="alert" aria-label="close">**&times;**</a>

57 <%= session.getAttribute("sucess").toString()%>

58 </div>

59 <%}

60 session.removeAttribute("sucess");%>

61

62

63 <table class="table table-striped table-responsive table-hover ">

64 <thead>

65 <tr class="info">

66 <th>Location ID</th>

67 <th>Location Name</th>

68 <th>Current Travel Leg Number</th>

69 <th>View</th>

70 <th>Add</th>

71 <th>Delete</th>

72 </tr>

73 </thead>

74 <tbody>

75 <% LocationDA locaDA = new LocationDA();

76 adjencyDA adDA = new adjencyDA();

77 List<Location> locaList = new List<Location>();

78 locaList = locaDA.getAllLocation();

79 for (int i = 0; i < locaList.size(); i++) {%>

80

81 <tr **ng-repeat**="user in users">

82

83 <td><%= locaList.getValueOf(i + 1).getCode\_num()%> </td>

84 <td><%= locaList.getValueOf(i + 1).getName()%></td>

85 <td><%= adDA.getNumberofTravelLeg(locaList.getValueOf(i + 1).getCode\_num())%></td>

86 <td><button type="button" class="btn btn-success view" data-toggle="modal" data-target="#View<%=locaList.getValueOf(i + 1).getCode\_num()%>" >View</button></td>

87 <td><button type="button" class=" btn btn-info editbtn" data-toggle="modal" data-target="#modaledit" **ng-click**="selectUser(user)" value="<%= locaList.getValueOf(i + 1).getCode\_num()%>" >Edit</button></td>

88

89 <td><button type="button" class="btn btn-danger delete" data-toggle="modal" data-target="#<%=locaList.getValueOf(i + 1).getCode\_num()%>" **ng-click**="selectUser(user)">Delete</button></td>

90

91 </tr>

92 <%}%>

93 </tbody>

94 </table>

95 </div>

96 </div>

97

98

99 <%for (int i = 0; i < locaList.size(); i++) {%>

100 <div class="modal fade" id="View<%= locaList.getValueOf(i + 1).getCode\_num()%>" role="dialog">

101 <div class="modal-dialog">

102 <div class="modal-content">

103 <div class="modal-header">

104 <h4 class="modal-title">View All Travel Leg of <%= locaList.getValueOf(i + 1).getName()%></h4>

105 </div>

106

107 <table class="table table-striped table-responsive table-hover ">

108 <thead>

109 <tr class="info">

110 <th>Location ID</th>

111 <th>Location Name</th>

112 </tr>

113 </thead>

114 <%

115 List<Location> viewAll = adDA.selectDeleteLocation(String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()), String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()));

116 for (int j = 0; j < viewAll.size(); j++) {

117 %>

118 <tr>

119 <td><%= viewAll.getValueOf(j + 1).getCode\_num()%></td>

120 <td><%= viewAll.getValueOf(j + 1).getName()%></td>

121 </tr>

122 <%}%>

123 </table>

124

125

126

127

128 <div class="modal-footer">

129 <button type="submit" class="btn btn-info pull-right" data-dismiss="modal" >Close</button>

130 </div>

131 </div>

132 </div>

133 </div>

134 <%}%>

135

136

137

138

139 <div class="modal fade" id="modaledit" role="dialog">

140 <div class="modal-dialog">

141 <div class="modal-content">

142 <div class="modal-header">

143 <h4 class="modal-title">Add New Travel Leg</h4>

144 <h5>Editing: <div id="sourceLoca"></div></h5>

145 </div>

146 <div class="modal-body">

147 <form class="form-horizontal" action="editServelet">

148 <input id="sourceCode" name="sourceCode" hidden>

149 <div class="form-group">

150 <label class="control-label col-md-2">Location Name</label>

151 <div class="col-md-10">

152 <input list="location" type="text" id="locaName" name="locaName" class=" form-control">

153 <datalist id="location">

154 <%for (int i = 0; i < locaList.size(); i++) {%>

155

156 <option value="<%= locaList.getValueOf(i + 1).getName()%>" >

157 <%}%>

158 </datalist>

159 </div>

160 </div>

161

162 <div class="form-group">

163 <label class="control-label col-md-2">Location Code</label>

164 <div class="col-md-10">

165 <input type="text" class=" form-control" id="locaCode" name="locaCode" readonly="true">

166 </div>

167 </div>

168

169 <div class="form-group">

170 <div class="col-md-2 col-md-offset-2">

171 <button type="submit" class="btn btn-info " name="button" value="update">ADD</button>

172 </div>

173 </div>

174 </form>

175 </div>

176 <div class="modal-footer">

177 <button type="submit" class="btn btn-info pull-right" data-dismiss="modal" >Close</button>

178 </div>

179 </div>

180 </div>

181 </div>

182

183

184

185 <%for (int i = 0; i < locaList.size(); i++) {%>

186 <form action="editServelet">

187 <div class="modal fade" id="<%= locaList.getValueOf(i + 1).getCode\_num()%>" role="dialog">

188 <div class="modal-dialog">

189 <div class="modal-content">

190 <div class="modal-header">

191 <h4 class="modal-title">Deleting Travel Leg</h4>

192 </div>

193 <div class="modal-body">

194 <input list="<%=locaList.getValueOf(i + 1).getName()%>" type="text" class=" form-control" name="delDest">

195 <datalist id="<%=locaList.getValueOf(i + 1).getName()%>">

196 <%

197 List<List<Location>> delLoca = new List<List<Location>>();

198 delLoca.add(adDA.selectDeleteLocation(String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num()), String.*valueOf*(locaList.getValueOf(i + 1).getCode\_num())));

199 for (int j = 0; j < delLoca.getValueOf(1).size(); j++) {

200 %>

201

202 <option value="<%= delLoca.getValueOf(1).getValueOf(j + 1).getName()%>">

203 <%}%>

204 </datalist>

205

206 <input value="<%= locaList.getValueOf(i + 1).getCode\_num()%>" name="delSource">

207

208 </div>

209 <div class="modal-footer">

210 <button type="submit" class="btn btn-info " >Yes</button>

211 <button type="button" class="btn btn-info " data-dismiss="modal" >No</button>

212

213 </div>

214 </div>

215 </div>

216 </div>

217 </form>

218 <%}%>

219 </body>

220 </html>

221 </body>

222 </html>

223 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

224 <script>

225

226

227

228

229 $("#locaName").change(function () {

230 var location = document.getElementById("locaName").value;

231

232 <% for (int i = 0; i < locaList.size(); i++) {%>

233 if (location == '<%= locaList.getValueOf(i + 1).getName()%>') {

234 $("#locaCode").val("<%= locaList.getValueOf(i + 1).getCode\_num()%>");

235 }

236 <%}%>

237 }

238 );

239

240 $(".editbtn").click(function (e) {

241 sourceCode = $(this).val();

242 <% for (int i = 0; i < locaList.size(); i++) {%>

243 if (sourceCode == '<%= locaList.getValueOf(i + 1).getCode\_num()%>') {

244 $("#sourceLoca").html("Name : <%= locaList.getValueOf(i + 1).getName()%> <br/> Location Code : <%= locaList.getValueOf(i + 1).getCode\_num()%>");

245 $("#sourceCode").val("<%= locaList.getValueOf(i + 1).getCode\_num()%>");

246 }

247 <%}%>

248

249

250 });

251

252

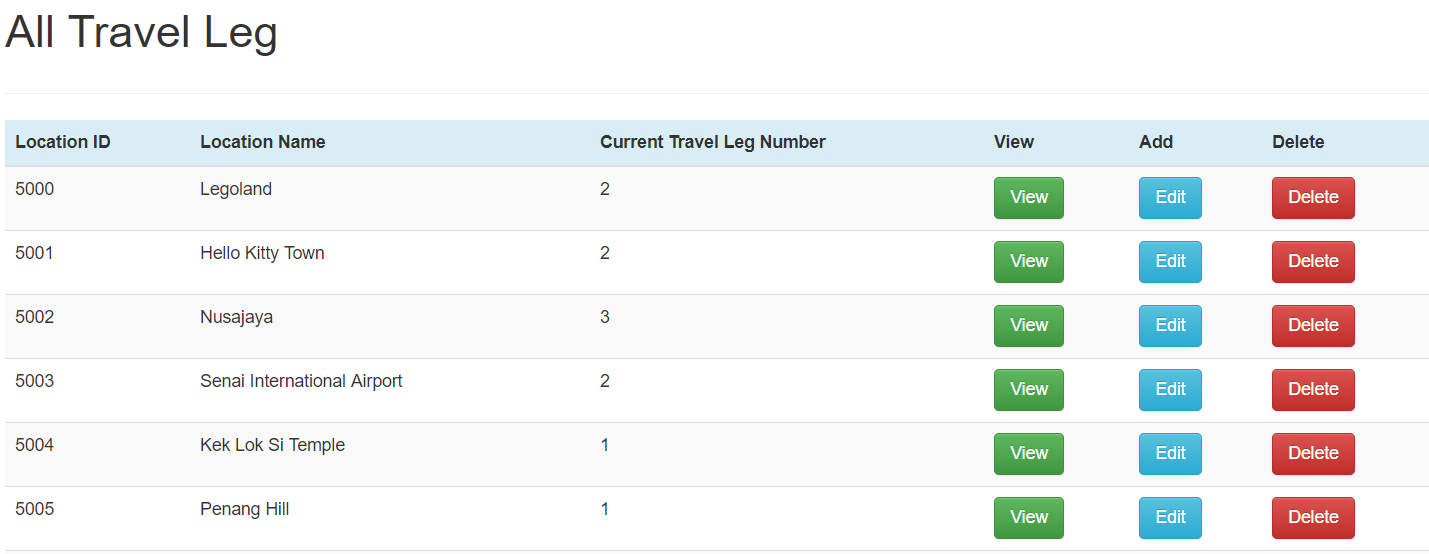
253

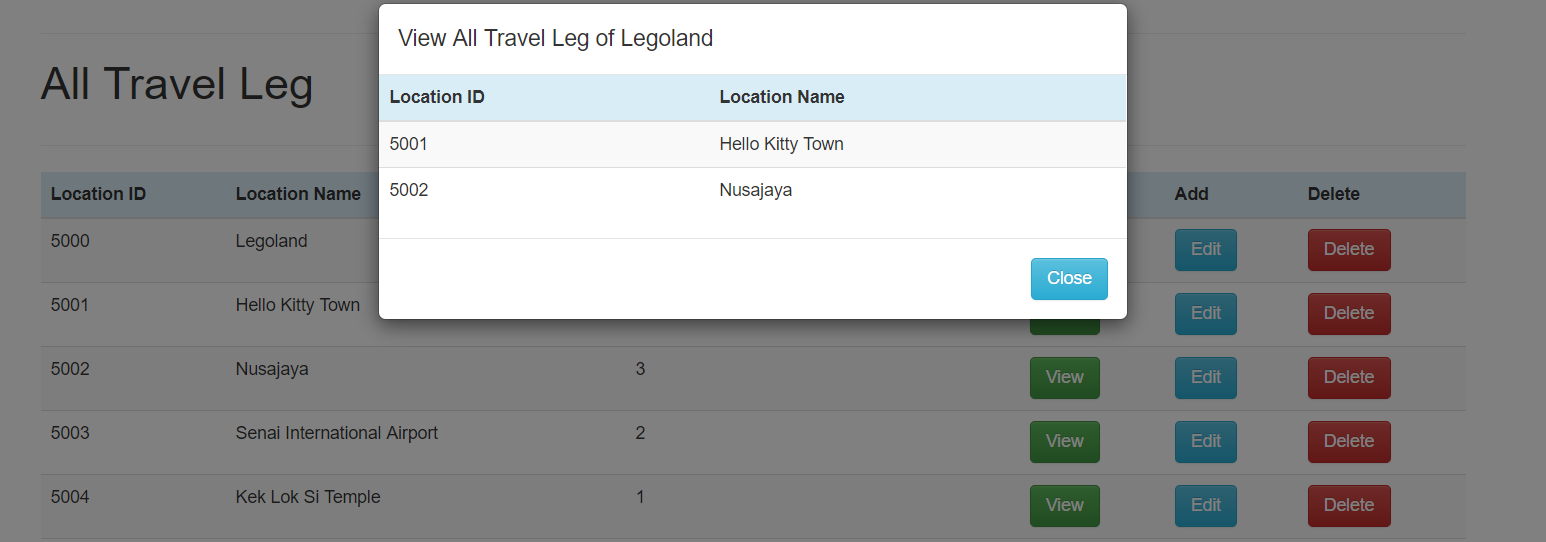
254

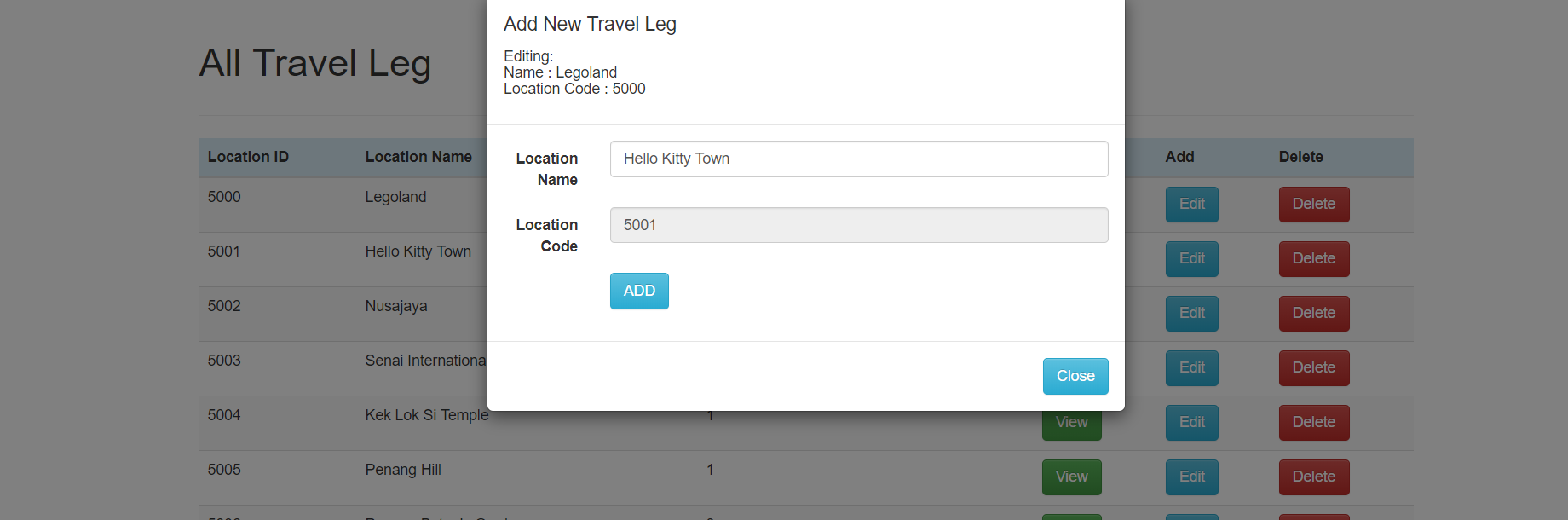
255

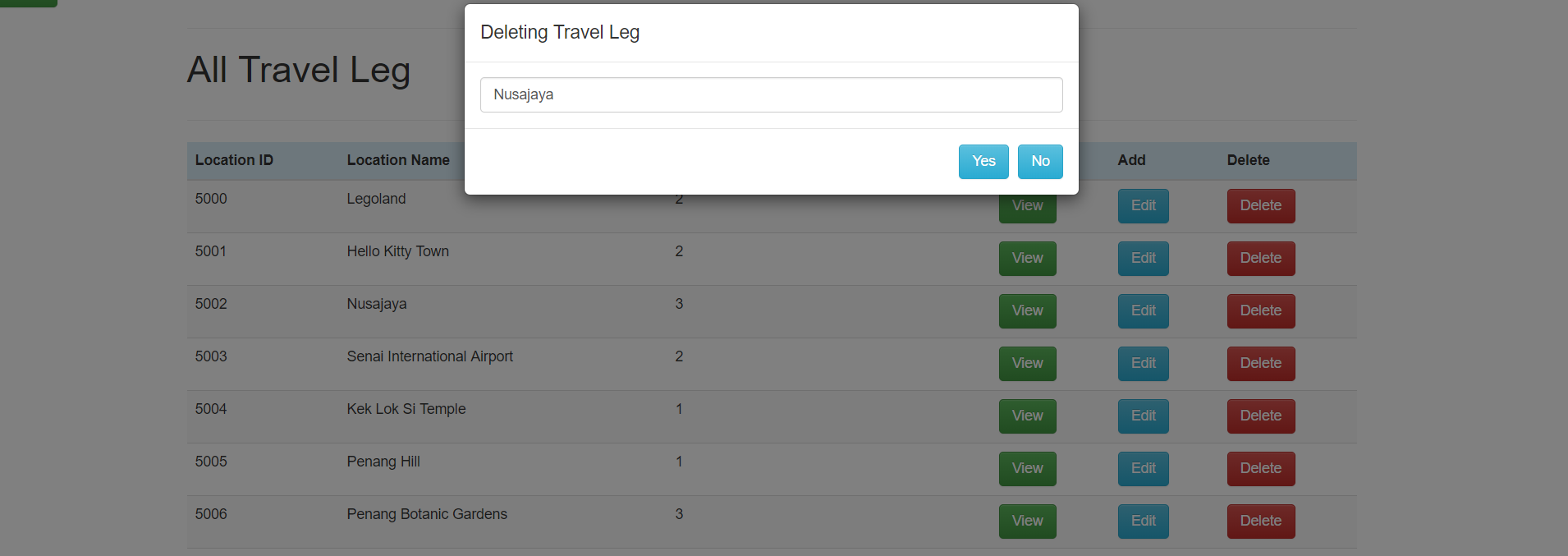
256 </script>

257









The figure above shows that the user allow to add travel leg, view travel leg and delete travel leg.

Name : Choong Mun Sin

**Distance Between Two Location**

Distance between two location or travel legs are calculated by the longitude and latitude of the location. We found that if we use the longitude and latitude to calculate the distance between two location will be just a straight line ignoring all the terrains in between hence we implement the A star algorithm and breath first search algorithm with our calculation. Each of the location acts as the “road” to the destination. For example, from point A to point F, the system must pass by point B,C,D,E instead of just calculating the distance by longitude and latitude between two points A and point F. The method will calculate the distance between point A to point B, point B to point C and so on and total up to get an accurate distance. The more location you have in the database the more accurate the distance will be.

Source Code :

1 /\*

2 \* To change this license header, choose License Headers in Project Properties.

3 \* To change this template file, choose Tools | Templates

4 \* and open the template in the editor.

5 \*/

6 package domain;

7

8 /\*\*

9 \*

10 \* **@author** User

11 \*/

12 public class **Location** {

13 private double lat;

14 private double lon;

15 private String name;

16 private int code\_num;

17 private char type; // s m l n p k

18 // private char geography ; // c s hb

19 private boolean t\_station;

20 private boolean b\_station;

21 private boolean flight ;

22 private String link;

23

24 private List <Location> adjencies = new List <Location>();

25 private double g\_score;

26 private double h\_score;

27 private double f\_score;

28 private Location parent;

29 private int upper;

30 private int adjency;

31

32 public **Location**() {

33 this.g\_score = 0;

34 this.h\_score = 0;

35 this.f\_score = 0;

36 this.parent = null;

37 code\_num= 0;

38 }

39 // for driver program testing

40 public **Location**(double lat, double lon, String name, char type/\*, char geography\*/) {

41 this();

42 this.lat = lat;

43 this.lon = lon;

44 this.name = name;

45 this.type = type;

46 // this.geography = geography;

47

48 }

49

50 public **Location**( int code\_num,String name) {

51 this.name = name;

52 this.code\_num = code\_num;

53 }

54

55 public **Location**(int code\_num, String name, int upper, String link) {

56 this.name = name;

57 this.code\_num = code\_num;

58 this.link = link;

59 this.upper = upper;

60 }

61

62

63 // real

64 public **Location**(double lat, double lon, String name, int code\_num, char type /\*, char geography\*/, boolean t\_station,boolean b\_station,boolean flight,int upper,String link) {

65 this.lat = lat;

66 this.lon = lon;

67 this.name = name;

68 this.code\_num = code\_num;

69 this.type = type;

70 // this.geography = geography;

71 this.flight = flight;

72 this.t\_station = t\_station;

73 this.b\_station = b\_station;

74 this.upper=upper;

75 this.link=link;

76 }

77 //yj

78 public **Location**(double lat, double lon, String name, int code\_num, char type , boolean t\_station,boolean b\_station,boolean flight,int upper,String link,int adjency) {

79 this.lat = lat;

80 this.lon = lon;

81 this.name = name;

82 this.code\_num = code\_num;

83 this.type = type;

84 this.flight = flight;

85 this.t\_station = t\_station;

86 this.b\_station = b\_station;

87 this.upper=upper;

88 this.link=link;

89 this.adjency=adjency;

90 }

91 public **Location**(Location loca){

92 this.lat = loca.getLat();

93 this.lon = loca.getLon();;

94 this.name = loca.getName();;

95 this.code\_num = loca.getCode\_num();;

96 this.type = loca.getType();;

97 this.t\_station = loca.isT\_station();;

98 this.b\_station = loca.isB\_station();;

99 this.flight = loca.isFlight();;

100 this.link = loca.getLink();;

101 this.adjency = loca.getAdjency();;

102 this.g\_score = loca.getG\_score();;

103 this.h\_score = loca.getH\_score();;

104 this.f\_score = loca.getF\_score();;

105 this.parent = loca.getParent();;

106 this.upper = loca.getUpper();;

107 }

108

109 public void **setLat**(double lat) {

110 this.lat = lat;

111 }

112

113 public void **setLon**(double lon) {

114 this.lon = lon;

115 }

116

117 public void **setName**(String name) {

118 this.name = name;

119 }

120

121 public void **setCode\_num**(int code\_num) {

122 this.code\_num = code\_num;

123 }

124

125

126 public void **setType**(char type) {

127 this.type = type;

128 }

129

130 // public void setGeography(char geography) {

131 // this.geography = geography;

132 // }

133

134 public void **setFlight**(boolean flight) {

135 this.flight = flight;

136 }

137

138

139

140 public void **setT\_station**(boolean t\_station) {

141 this.t\_station = t\_station;

142 }

143

144 public void **setB\_station**(boolean b\_station) {

145 this.b\_station = b\_station;

146 }

147

148 public void **setUpper**(int upper) {

149 this.upper = upper;

150 }

151

152 public void **setAdjencies**(List<Location> adjencies) {

153 this.adjencies = adjencies;

154 }

155

156 public void **setG\_score**(double g\_score) {

157 this.g\_score = g\_score;

158 }

159

160 public void **setH\_score**(double h\_score) {

161 this.h\_score = h\_score;

162 }

163

164 public void **setF\_score**(double f\_score) {

165 this.f\_score = f\_score;

166 }

167

168 public void **setParent**(Location parent) {

169 this.parent = parent;

170 }

171

172

173 public static void ***starsearch***(Location start , Location end){

174 List<Location> explored = new List<Location>();

175 List<Location> openset = new List<Location>();

176 openset.add(start);

177

178 while(openset.size()>0){

179 Location current = openset.getValueOf(1);

180 for(int i = 0 ; i<openset.size();i++){

181 if(openset.getValueOf(i+1).getF\_score() < current.getF\_score() ||openset.getValueOf(i+1).getF\_score() == current.getF\_score() && openset.getValueOf(i+1).getH\_score() < current.getH\_score() )

182 current = openset.getValueOf(i+1);

183 }

184 openset.remove(current);

185 explored.add(current);

186 if(current == end){

187 return;

188 }

189

190 for( int i = 0 ; i < current.getAdjencies().size() ;i++)

191 {

192 Location neighbour = current.getAdjencies().getValueOf(i+1);

193 if(explored.contain(neighbour))

194 continue;

195 double new\_f\_cost=current.getG\_score() + neighbour.distance(current);

196 if(new\_f\_cost<neighbour.getG\_score()|| !openset.contain(neighbour)){

197 neighbour.setG\_score(new\_f\_cost);

198 neighbour.setH\_score(neighbour.distance(end));

199 neighbour.setParent(current);

200

201 if(!openset.contain(neighbour))

202 openset.add(neighbour);

203 }

204

205

206 }

207

208 }

209 }

210

211

212

213 public double **distance**(Location endlo) {

214 if ((lat == endlo.getLat()) && (lon == endlo.getLon())) {

215 return 0;

216 } else {

217

218 double theta = this.lon - endlo.getLon();

219 double dist = Math.*sin*(Math.*toRadians*(lat)) \* Math.*sin*(Math.*toRadians*(endlo.getLat())) + Math.*cos*(Math.*toRadians*(lat)) \* Math.*cos*(Math.*toRadians*(endlo.getLat())) \* Math.*cos*(Math.*toRadians*(theta));

220 dist = Math.*acos*(dist);

221 dist = Math.*toDegrees*(dist);

222 dist = dist \* 60 \* 1.1515;

223 //if (unit == "K") {

224 dist = dist \* 1.609344;

225 // } else if (unit == "N") {

226 // dist = dist \* 0.8684;

227 // }

228

229 return (dist);

230 }

231 }

232 public boolean **equalcode**(int code){

233 return this.code\_num==code;

234 }

235

236 public static double ***adjust***(Location start, Location end , linkedlist<Location> result\_link\_loc) {

237 LinkedStack<Location> stack\_loc = new LinkedStack<>();

238 double cost = 0;

239 System.*out*.println("/");

240 System.*out*.println(start.toString());

241 System.*out*.println(end.toString());

242 System.*out*.println("/");

243 // System.out.println(end.getName());

244 for (Location node = end; node != start; node = node.getParent()) {

245 stack\_loc.push(node);

246 // System.out.println(node.toString());

247 }

248 int size = stack\_loc.getCurrsize();

249 for(int i = 0 ; i < size;i++){

250 // System.out.println(stack\_loc.getCurrsize());

251 result\_link\_loc.add(stack\_loc.pop());

252 }

253 return cost;

254 }

255 public static double ***print***(Location start, Location end) {

256 double cost = 0;

257 System.*out*.println(end.getName());

258 for (Location node = end; node != start; node = node.getParent()) {

259 System.*out*.println(node.getParent().getName());

260 cost += node.distance(node.getParent());

261 }

262 return cost;

263 }

264

265 public double **getLat**() {

266 return lat;

267 }

268

269 public double **getLon**() {

270 return lon;

271 }

272

273 public String **getName**() {

274 return name;

275 }

276

277 public int **getCode\_num**() {

278 return code\_num;

279 }

280

281 public int **getUpper**() {

282 return upper;

283 }

284

285 public boolean **isT\_station**() {

286 return t\_station;

287 }

288

289 public boolean **isB\_station**() {

290 return b\_station;

291 }

292

293 public boolean **isFlight**() {

294 return flight;

295 }

296

297 public int **getAdjency**() {

298 return adjency;

299 }

300

301 public char **getType**() {

302 return type;

303 }

304

305 // public char getGeography() {

306 // return geography;

307 // }

308

309 public List<Location> **getAdjencies**() {

310 return adjencies;

311 }

312

313 public double **getG\_score**() {

314 return g\_score;

315 }

316

317 public double **getH\_score**() {

318 return h\_score;

319 }

320

321 public double **getF\_score**() {

322 return f\_score;

323 }

324

325 public Location **getParent**() {

326 return parent;

327 }

328

329 public String **getLink**() {

330 return link;

331 }

332

333 @Override

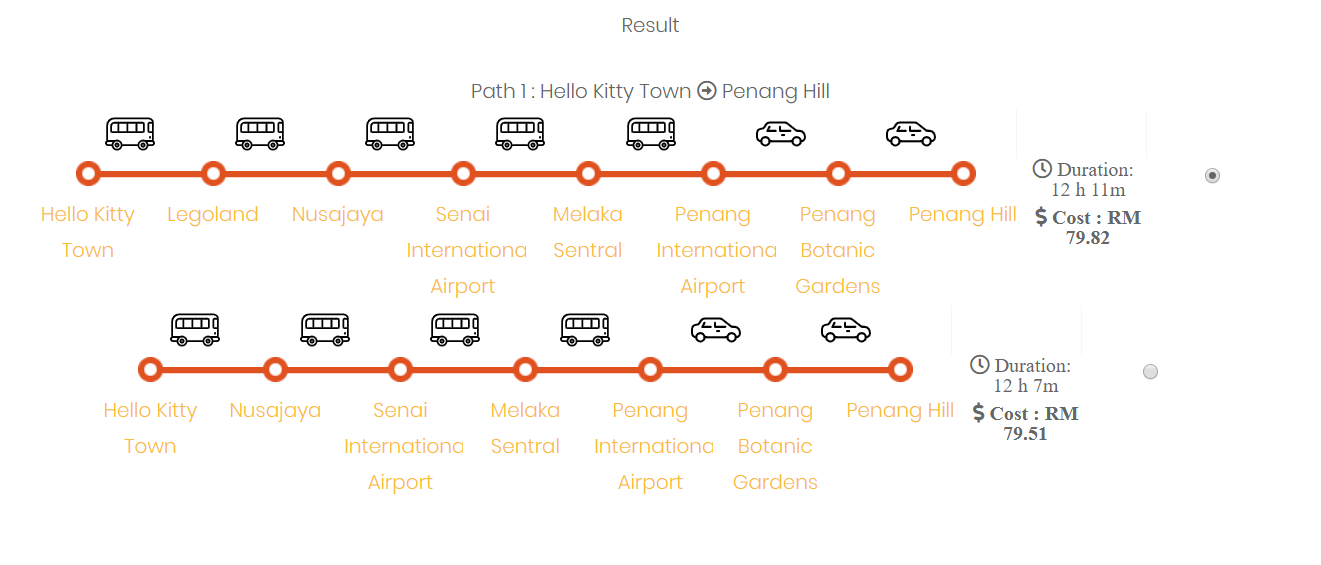
334 public String **toString**() {

335 return "Location{" + "lat=" + lat + ", lon=" + lon + ", name=" + name + ", code\_num=" + code\_num + ", type=" + type + /\*", geography=" + geography +\*/ ", flight=" + flight + ", station=" + t\_station + '}' +"" ;

336 }

337

338 }



This figure above shows price differently because of the different distance.

**Location Details**

Every location detail will be retrieved from the database and store it in an ADT before processing it, this is to reduce the numbers of accessing to the database hence increasing the efficiency of the overall system. The CRUD methods will then use the location data in the ADT to process.

Source Code :

6 package Servlet;

7

8 import DA.LocationDA;

9 import DA.categoriesDA;

10 import domain.List;

11 import domain.Location;

12 //import domain.linkedlist;

13 import java.io.IOException;

14 import java.io.PrintWriter;

15 import javax.servlet.ServletException;

16 import javax.servlet.http.HttpServlet;

17 import javax.servlet.http.HttpServletRequest;

18 import javax.servlet.http.HttpServletResponse;

19 import javax.servlet.http.HttpSession;

20

21 /\*\*

22 \*

23 \* **@author** User

24 \*/

25 public class **DisplayServlet** extends HttpServlet {

26

27 protected void **processRequest**(HttpServletRequest request, HttpServletResponse response)

28 throws ServletException, IOException {

29 HttpSession s = request.getSession();

30

31 response.setContentType("text/html;charset=UTF-8");

32 String submit = request.getParameter("submit");

33 String src = request.getParameter("country\_id");

34 String reload = request.getParameter("reload");

35 String display = request.getParameter("display");

36 String edit = request.getParameter("edit");

37 String delete = request.getParameter("delete");

38 String locationtype = request.getParameter("locationtpye");

39 LocationDA lda = new LocationDA();

40 categoriesDA cda = new categoriesDA();

41 List<Location> loc = new List<Location>();

42

43 try (PrintWriter out = response.getWriter()) {

44

45 /\* TODO output your page here. You may use following sample code. \*/

46 if (reload != null) {

47 loc = cda.getAllcont();

48 for\_printing\_sub\_place(loc, out);

49

50 //System.out.println(6);

51 } else if (src != null) {

52 if (src.charAt(0) == '1') { // next btn funtion

53 loc = cda.getSomecountry(src);

54 for\_printing\_sub\_place(loc, out);

55

56 } else if (src.charAt(0) == '2') {

57

58 loc = cda.getSomestate(src);

59 for\_printing\_sub\_place(loc, out);

60

61 } else if (src.charAt(0) == '3') {

62 System.*out*.println(src);

63 loc = lda.getsomeLocation(3, src);

64

65 for (int i = 0; i < loc.size(); i++) {

66 out.println("<form action=**\"**staff\_function.jsp**\"** action=**\"**post**\"** ><div class=**\"**col-md-3 col-sm-6**\"**>**\n**"

67 + " <div class=**\"**product-grid6**\"**>**\n**"

68 + " <div class=**\"**product-image6**\"**>**\n**"

69 + " <a href=**\"**#**\"** >**\n**"

70 + " <img class=**\"**pic-1**\"** src=**\"**" + loc.getValueOf(i + 1).getLink() + "**\"** >**\n**"

71 + " </a>**\n**"

72 + " **\n**"

73 + " </div>**\n**"

74 + " <div class=**\"**product-content**\"**>**\n**"

75 + " <h3 class=**\"**title**\"**><a href=**\"**#**\"**> " + loc.getValueOf(i + 1).getName() + "</a></h3>**\n**"

76 + " **\n**"

77 + " </div>**\n**"

78 + " **\n**"

79 + " <ul class=**\"**social**\"**>**\n**"

80 + " <li><button name=**\"**display**\"** class=**\"**Display btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Display</button></li>**\n**"

81 + " <li><button name=**\"**edit**\"** class=**\"**Edit btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Edit</button></li>**\n**"

82 + " <li><button name=**\"**delete**\"** class=**\"**Delete btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Delete</button></li>**\n**"

83 + "**\n**"

84 + " </ul>**\n**"

85 + " </div>**\n**"

86 + " </div> </form>");

87 }

88

89 }

90 } else if (display != null) {

91 if (display.charAt(0) == '1') { // display btn funtion

92 System.*out*.println(display);

93 loc = lda.getsomeLocation(4, display);

94 for\_printing\_sub\_place(loc, out);

95

96 } else if (display.charAt(0) == '2') { // display btn funtion

97 System.*out*.println(display);

98 loc = lda.getsomeLocation(5, display);

99 for\_printing\_sub\_place(loc, out);

100

101 } else if (display.charAt(0) == '3') { // display btn funtion

102 System.*out*.println(display);

103 loc = lda.getsomeLocation(6, display);

104 for\_printing\_sub\_place(loc, out);

105

106 } else if (display.charAt(0) == '5') { // display btn funtion

107 Location l = lda.get\_one\_loc(1, display);

108 s.setAttribute("display", l);

109 response.sendRedirect("staff\_function.jsp");

110 }

111 } else if (edit != null) {

112 Location l = null;

113 if (edit.charAt(0) == '1') { // edit btn funtion

114 l = cda.get\_one\_Continent\_detail(edit);

115 } else if (edit.charAt(0) == '2') { // edit btn funtion

116 l = cda.get\_one\_Country\_detail(edit);

117 } else if (edit.charAt(0) == '3') { // edit btn funtion

118 l = cda.get\_one\_State\_detail(edit);

119 } else if (edit.charAt(0) == '5') { // edit btn funtion

120 l = lda.get\_one\_loc(1, edit);

121 }

122 s.setAttribute("edit", l);

123 response.sendRedirect("staff\_function.jsp");

124

125 }

126 else if (delete != null) {

127 Location l = null;

128 if (delete.charAt(0) == '1') { // edit btn funtion

129 l = cda.get\_one\_Continent\_detail(delete);

130 } else if (delete.charAt(0) == '2') { // edit btn funtion

131 l = cda.get\_one\_Country\_detail(delete);

132 } else if (delete.charAt(0) == '3') { // edit btn funtion

133 l = cda.get\_one\_State\_detail(delete);

134 } else if (delete.charAt(0) == '5') { // edit btn funtion

135 l = lda.get\_one\_loc(1, delete);

136 }

137 s.setAttribute("delete", l);

138 response.sendRedirect("staff\_function.jsp");

139

140 }

141 else if (locationtype != null){

142

143 loc = lda.getsomeLocation(1, locationtype);

144 for\_printing\_sub\_place(loc, out);

145 }

146

147 }

148

149 }

150

151 public void **for\_printing\_sub\_place**(List<Location> loc, PrintWriter out) {

152

153 for (int i = 0; i < loc.size(); i++) {

154 out.println("<div class=**\"**col-md-3 col-sm-6**\"**>**\n**"

155 + " <div class=**\"**product-grid6**\"**>**\n**"

156 + " <div class=**\"**product-image6**\"**>**\n**"

157 + " <a href=**\"**#**\"** >**\n**"

158 + " <img class=**\"**pic-1**\"** alt=**\"\"** src=**\"** " + loc.getValueOf(i + 1).getLink() + " **\"** >**\n**"

159 + " </a>**\n**"

160 + " **\n**"

161 + " </div>**\n**"

162 + " <div class=**\"**product-content**\"**>**\n**"

163 + " <h3 class=**\"**title**\"**><a href=**\"**#**\"**> " + loc.getValueOf(i + 1).getName() + "</a></h3>**\n**"

164 + " **\n**"

165 + " </div>**\n**"

166 + " **\n**"

167 + " <ul class=**\"**social**\"**>**\n**"

168 + " <li><button class=**\"**country\_list btn btn-primary**\"** id=**\"**country\_list**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > next</button></li>**\n**"

169 + " <li><button name=**\"**display**\"** class=**\"**Display btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Display</button></li>**\n**"

170 + " <li><button name=**\"**edit**\"** class=**\"**Edit btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Edit</button></li>**\n**"

171 + " <li><button name=**\"**delete**\"** class=**\"**Delete btn btn-primary**\"** value=**\"**" + loc.getValueOf(i + 1).getCode\_num() + "**\"** > Delete</button></li>**\n**"

172 + " **\n**"

173 + " </ul>**\n**"

174 + " </div>**\n**"

175 + " </div>");

176

177 }

178

179 out.println("<script type=**\"**text/javascript**\"** class=**\"**remove**\"**>**\n**"

180 + " **\n**"

181 + " **\n**"

182 + " $(**\"**.country\_list**\"**).click(function (e) {**\n**"

183 + " var x = $(this).val();**\n**"

184 + " e.preventDefault();**\n**"

185 + " $.ajax({**\n**"

186 + " url: **\"**DisplayServlet**\"**,**\n**"

187 + " // type: **\"**post**\"**,**\n**"

188 + " data: {country\_id: x},**\n**"

189 + " cache: false,**\n**"

190 + " success: function(data) {**\n**"

191 + " $(**\"**.row**\"**).html(data).slideDown('slow');**\n**"

192 + " }**\n**"

193 + " });**\n**"

194 + " $(**\"**.remove**\"**).remove();**\n**"

195 + "});**\n**"

196 + "**\n**"

197 + "$(**\"**.Display**\"**).click(function (e) {**\n**"

198 + " var x = $(this).val();**\n**"

199 + " e.preventDefault();**\n**"

200 + " $.ajax({**\n**"

201 + " url: **\"**DisplayServlet**\"**,**\n**"

202 + " // type: **\"**post**\"**,**\n**"

203 + " data: {display: x},**\n**"

204 + " cache: false,**\n**"

205 + " success: function(data) {**\n**"

206 + " $(**\"**.row**\"**).html(data).slideDown('slow');**\n**"

207 + " }**\n**"

208 + " });**\n**"

209 + " $(**\"**.remove**\"**).remove();**\n**"

210 + "});**\n**"

211 + "**\n**"

212 + "$(**\"**.Edit**\"**).click(function (e) {**\n**"

213 + " var x = $(this).val();**\n**"

214 + " e.preventDefault();**\n**"

215 + " $.ajax({**\n**"

216 + " url: **\"**DisplayServlet**\"**,**\n**"

217 + " // type: **\"**post**\"**,**\n**"

218 + " data: {edit: x},**\n**"

219 + " cache: false,**\n**"

220 + " success: function(data) {**\n**"

221 + " $(**\"**.row**\"**).html(data).slideDown('slow');**\n**"

222 + " }**\n**"

223 + " });**\n**"

224 + " $(**\"**.remove**\"**).remove();**\n**"

225 + "});**\n**"

226 + "**\n**"

227 + "$(**\"**.Delete**\"**).click(function (e) {**\n**"

228 + " var x = $(this).val();**\n**"

229 + " e.preventDefault();**\n**"

230 + " $.ajax({**\n**"

231 + " url: **\"**DisplayServlet**\"**,**\n**"

232 + " // type: **\"**post**\"**,**\n**"

233 + " data: {delete : x},**\n**"

234 + " cache: false,**\n**"

235 + " success: function(data) {**\n**"

236 + " $(**\"**.row**\"**).html(data).slideDown('slow');**\n**"

237 + " }**\n**"

238 + " });**\n**"

239 + " $(**\"**.remove**\"**).remove();**\n**"

240 + "});**\n**"

241 + "</script>");

242

243 }

244

245 @Override

246 protected void **doGet**(HttpServletRequest request, HttpServletResponse response)

247 throws ServletException, IOException {

248 processRequest(request, response);

249 }

250

251 @Override

252 protected void **doPost**(HttpServletRequest request, HttpServletResponse response)

253 throws ServletException, IOException {

254 processRequest(request, response);

255 }

256

257 @Override

258 public String **getServletInfo**() {

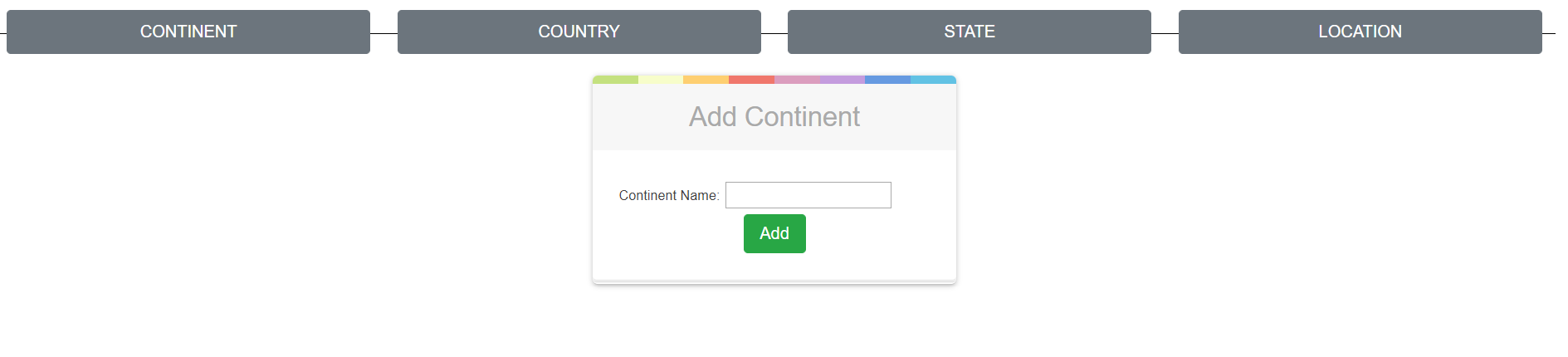
259 return "Short description";

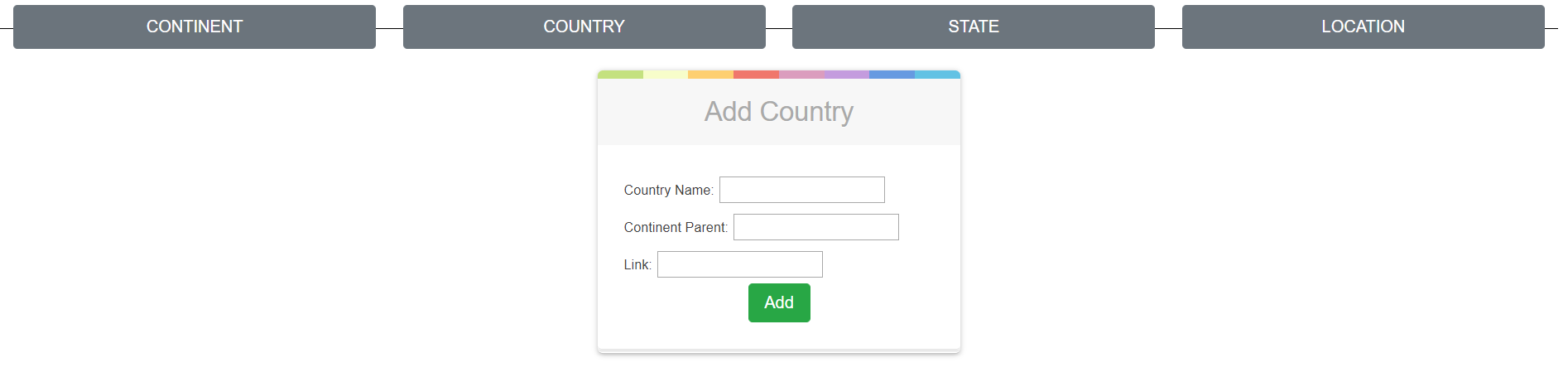
260 }// </editor-fold>

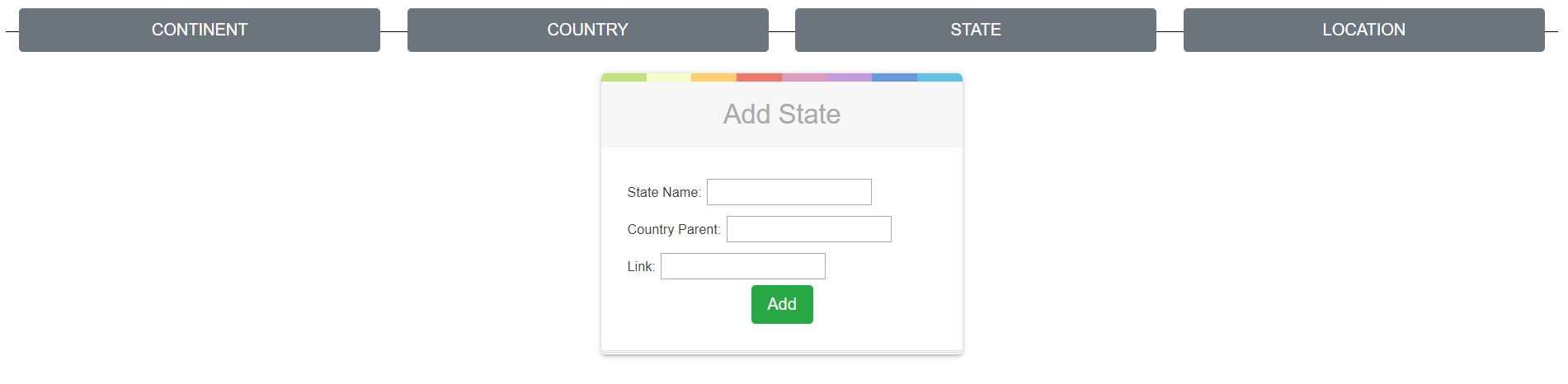
261

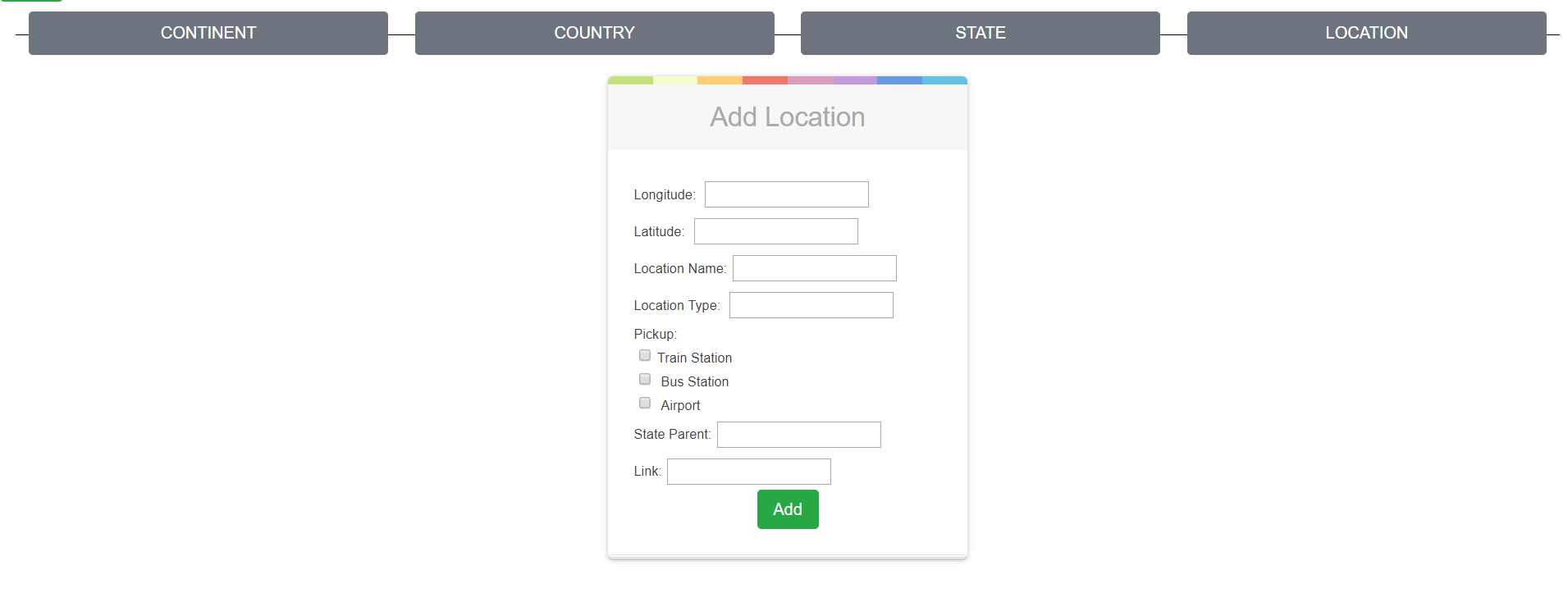
262 }

263

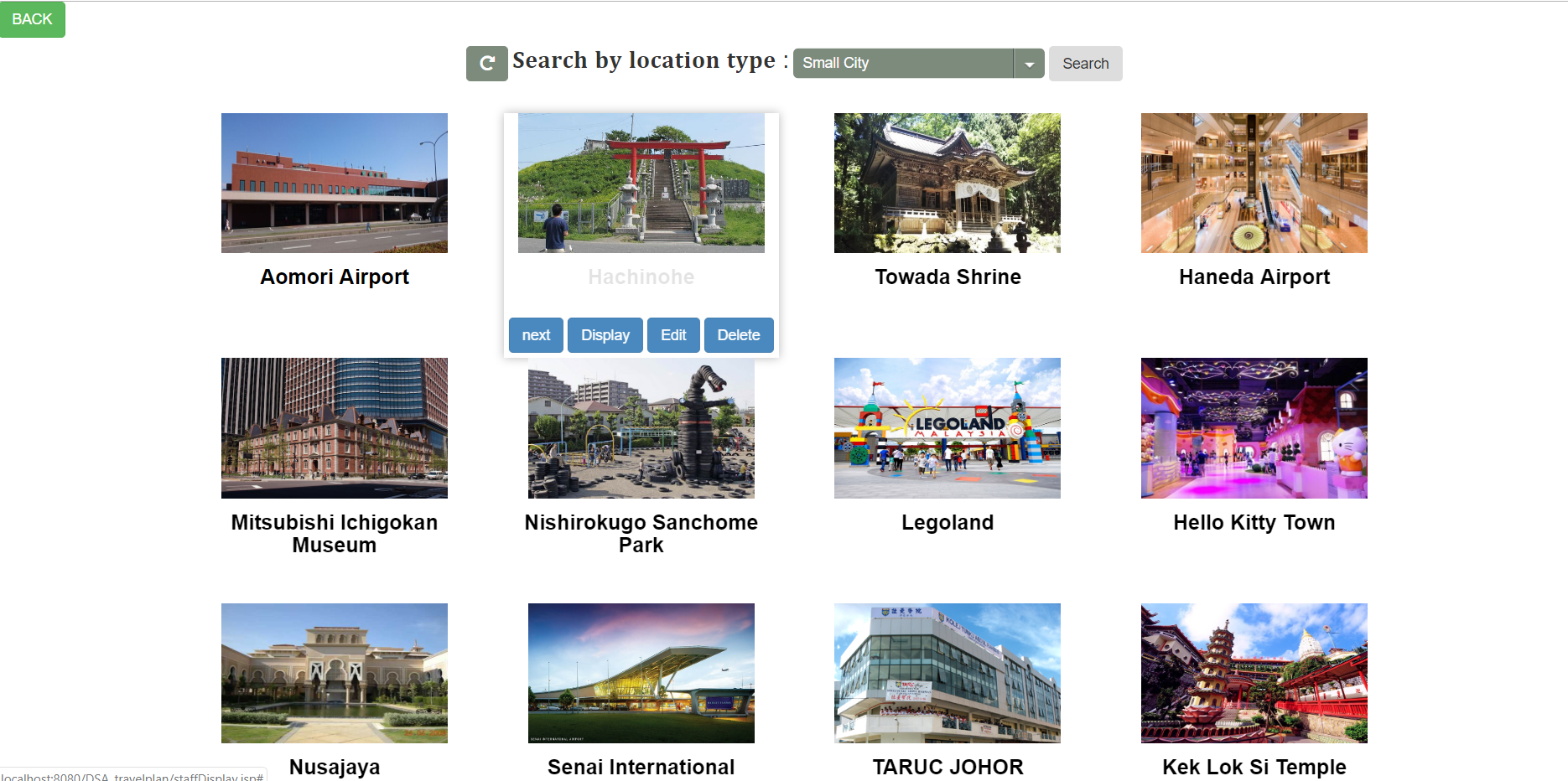








The figure above shows that the user is able to add new location into the database.



The figure above shows that the user is able to display all the location and edit or delete the location.