A Mini Project Report On

SMART DISPATCHER

Submitted in partial fulfilment of the requirements for the award of

VI Sem of Bachelor of Engineering

in

Computer Science And Engineering

By

G.Satvik Kalyan

160116733106

B.Sharath Chandra

160116733107

Under the guidance of

Mr G Vivek

Asst. Professor

Department of CSE

CBIT, Hyderabad

Dr. Ravinder Reddy Ramasani

Associate Professor

Department of CSE

CBIT, Hyderabad

Department of Computer Science and Engineering
Chaitanya Bharathi Institute of Technology(A)
Hyderabad - 500075
March, 2019

CERTIFICATE

This is to certify that the project work entitled 'Smart Dispatcher' submitted by G Satvik Kalyan(160116733106) and B Sharath Chandra(160116733107) in partial fulfillment of requirements for the award of degree of Bachelor of Engineering in Computer Science and Engineering as specialization is a record of the bonafide work carried out under the supervision of and this has not been submitted to any other university or institute for award of degree or diploma.

Mentors:

Mr G Vivek, Asst. Professor

Dr. Ravinder Reddy Ramasani ,Associate Professor

Head of Department

M Swamy Das

Professor and Head

Department of CSE

Batch Incharge

Mrs.Navdeep Kaur

Asst. Professor

Department of CSE,CBIT

DECLARATION

We, G Satvik Kalyan and B Sharath Chandra, hereby declare that the research work entitled "Smart Dispatcher" is original and bonafide work carried out by us as a part of fulfilment of Bachelor of Technology in Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Gandipet, Hyderabad. This project was done under the guidance of Mrs. Navdeep Kaur, Associate Professor, Dept. of CSE, CBIT. No part of the project work is copied from books/journals/internet and whichever sources the content can be borrowed from. The report is based on the project work done entirely by us and not copied from any other source.

G.Satvik Kalyan
160116733106
B.Sharath Chandra
160116733107

Abstract

In the real-world scenario e-commerce companies such as Amazon, Flipkart needs to deliver their items to the customers through the delivery parties. These delivery parties may not have an idea about the shortest path which they need to follow in order to minimize the distance, time and also money, saving unnecessary expenses.

This application aims at finding a path the party needs to travel such that he can deliver the items in a minimum distance. We give the locations of the customer and the distance between them, as the input, then application gives the path which he can follow to reach all the customers in a minimum distance. There can be constraints on the customers such as ‘customer availabity’ and then the path is determined accordingly.

List of Figures

Contents

1.Introduction

1.1 Objective

In this project, we find a path which the user needs to follow in order to cover all the destination in a minimum distance.

1.2 Problem Definition

Delivery executive picks a random order and delivers that order because of which there is a wastage of time and leads to unnecessary expenses.
Delivery parties may not have an idea about the shortest path which they need to follow in order to minimize the distance, time and also money.

1.3 Existed System

The already existing applications just randamly shows the user the customer addresses the user must manually think of a route for minimum distance path.

1.4 Proposed System

This application aims at finding a path the party needs to travel such that he can deliver the items in a minimum distance.Automated route planning that considers real-life constraints. Constraints such as customer availability are considered and the path is determined accordingly.

2. Literature Survey

In the real-world scenario e-commerce companies such as Amazon, Flipkart needs to deliver their items to the customers through the delivery parties. These delivery parties deliver these items randomly without having knowledge of the distance travelled. Sometimes delivering the items becomes difficult because of unavailablity of customers. The customers may not be available at that particular time when the delivery party reaches the location. Such conditions may lead to spending resources unnecessarily.

In order to minimise such expenses we need to think of a solution in which we can find an optimum route such that it covers all the customers as well as it takes minimum distance saving time as well as fuel. We need to compute this path considering constraints such as customer availability that is when ever the delivery party is going to start the delivery run in case he comes to know by some means that a particular customer may not be avaliable for that day. Then a shortest route must be calculated taking excluding that particular customer. Another such constraint is time constraint that is a customer may not be available for a particular duration of time. So if the delivery executive is informed about the unavailablility then the they can deliver to that particular afer that tim duration is lapsed.

Calculating the best possbile route of minimum distance considering all these constraints can be found out by applying travelling salesman algorithm which is a modification to Hamiltoninan cycle problem. The Hamiltoninan cycle problem is to find if there exist a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle [5].

We have to apply Travelling salesperson algorithm for a given set of delivery addresses in such a way that in case there is a constraint as mentioned above.A distance matrix is generated that is the distance between each and every node is calculated and given as an input to the algorithm along with the constraints. We then run the algorithm without considering these constrained locations.

3. Methodology (Design and implementation)

3.1 System Design

Our System is an isolated android application . Initially the user needs to login with valid credentials at the login screen.After successful login the user needs to select locations by dropping markers on google maps which uses google's maps api.After the user selects places the shortest path is found using travelling salesperson algorithm.Then the user can add constraints such as customer availability and time after which customer is available depending on the constraints the shortest path is again computed.

3.1.1 Proposed Algorithm

We use travelling sales person algorithm to find the best path from a set of given locations.
Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.The Following are different solutions for the traveling salesman problem.
Naive Solution:

- Consider city 1 as the starting and ending point.
- Generate all (n-1)! Permutations of cities.
- Calculate cost of every permutation and keep track of minimum cost permutation.
- Return the permutation with minimum cost.

Dynamic Programming:

Let the given set of vertices be {1, 2, 3, 4,….n}. Let us consider 1 as starting and ending point of output. For every other vertex i (other than 1), we find the minimum cost path with 1 as the starting point, i as the ending point and all vertices appearing exactly once. Let the cost of this path be cost(i), the cost of corresponding Cycle would be cost(i) + dist(i, 1) where dist(i, 1) is the distance from i to 1. Finally, we return the minimum of all [cost(i) + dist(i, 1)] values. This looks simple so far. Now the question is how to get cost(i)?
To calculate cost(i) using Dynamic Programming, we need to have some recursive relation in terms of sub-problems. Let us define a term *C*(S, i) be the cost of the minimum cost path visiting each vertex in set S exactly once, starting at 1 and ending at i.
We start with all subsets of size 2 and calculate C(S, i) for all subsets where S is the subset, then we calculate C(S, i) for all subsets S of size 3 and so on. Note that 1 must be present in every subset.

3.1.2 Diagrammatic Representation

Block Diagram

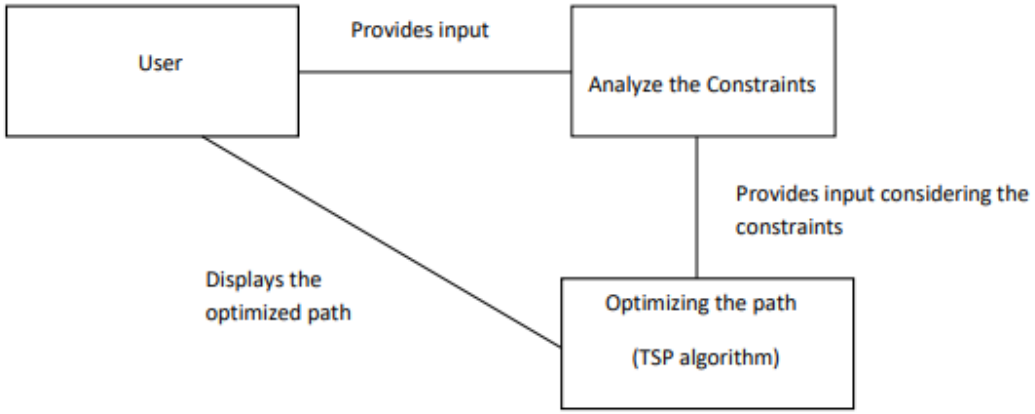


Fig-(1).Block Diagram

The user provides the locations as well as constraints. The travelling salesperson algorithm is applied for the given set of locations considering the constraints.

Activity Diagram

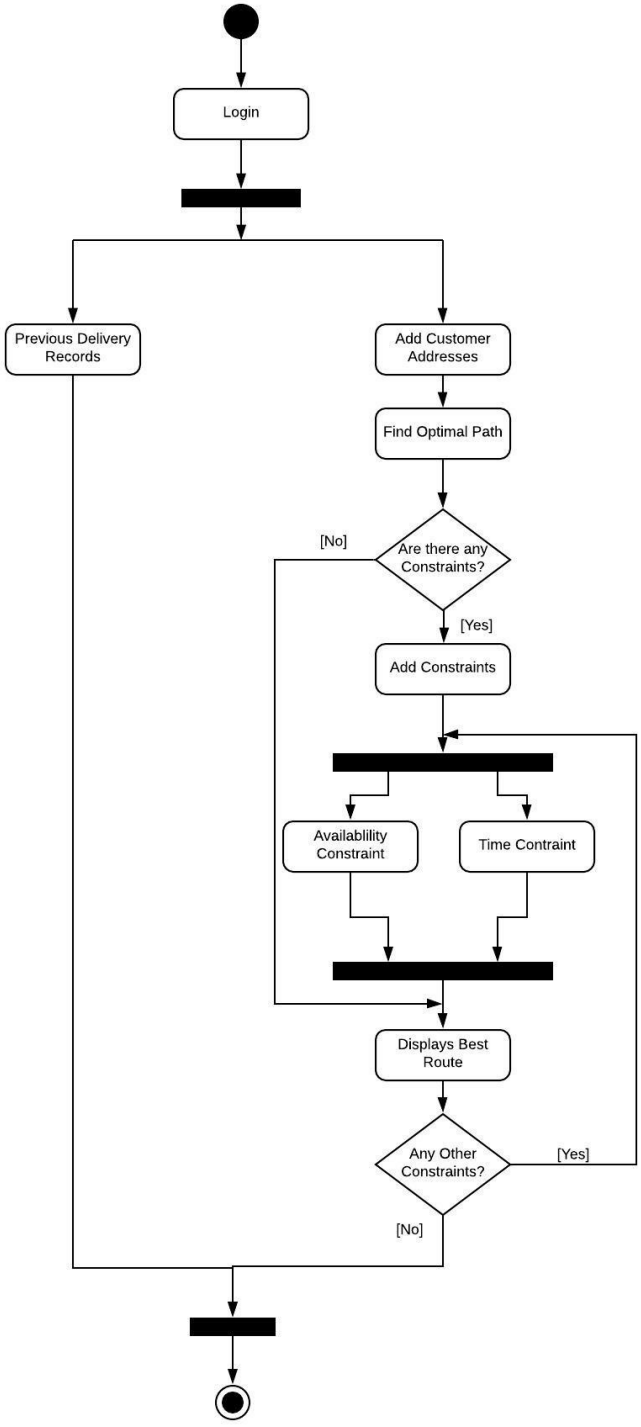


Fig-(2). Activity Diagram

In case the user provides any constraints such as availabity then that particular location is excluded from the distance matrix and again TSP algorithm is applied and new path is displayed from the current position.

In case time constraint is applied then that particular node is omitted for that particular amount of time after that mentioned time is lapsed the node gets activated and again TSP algorithm is applied considering that node and also the remaining nodes which are to be explored.

3.2 Implementation of Proposed solution

The user needs to place markers of delivery locations on maps using maps API which are considered as the inputs. Distance between every node is calculated and a distance matrix is generated for all the locations. Travelling salesperson algorithm is then applied for these locations.

An array of distances is created for every possible path and minimum of these distances is found out and the path is determined.

Implementation Flow

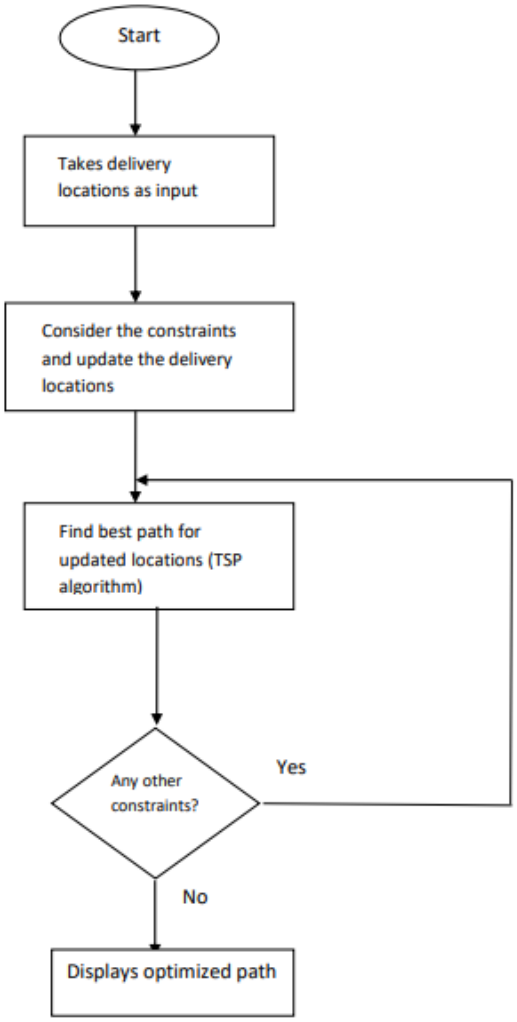


Fig-(3). Flow Diagram

The user gives delivery locations as an input to the application and then the optimized shortest path is displayed to the user. In case if there is a constraint such as the availability of the customer and available time of the customer they can be applied and the shortest path is obtained based on the constraints. These constraints can be applied at any point of time while delivering between locations.

3.3 System Requirements

Software requirements

- Android Studio
- Minimum SDK version 26
- Google Play Services
- Google Maps
- Internet Connectivity

Hardware requirements

- Android Mobile with minimum 2GB RAM.

4. Results and discussions

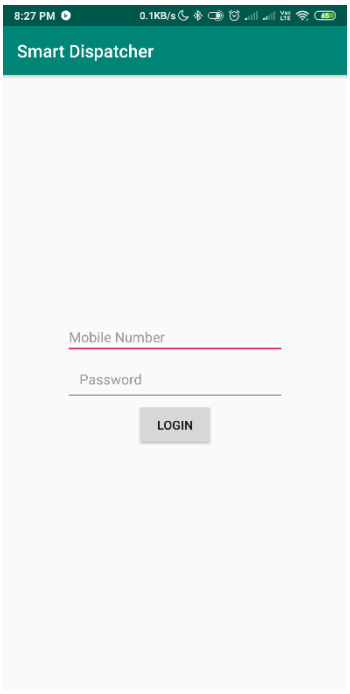


Fig-(4) : Login Screen

Fig-(5) : Home Screen

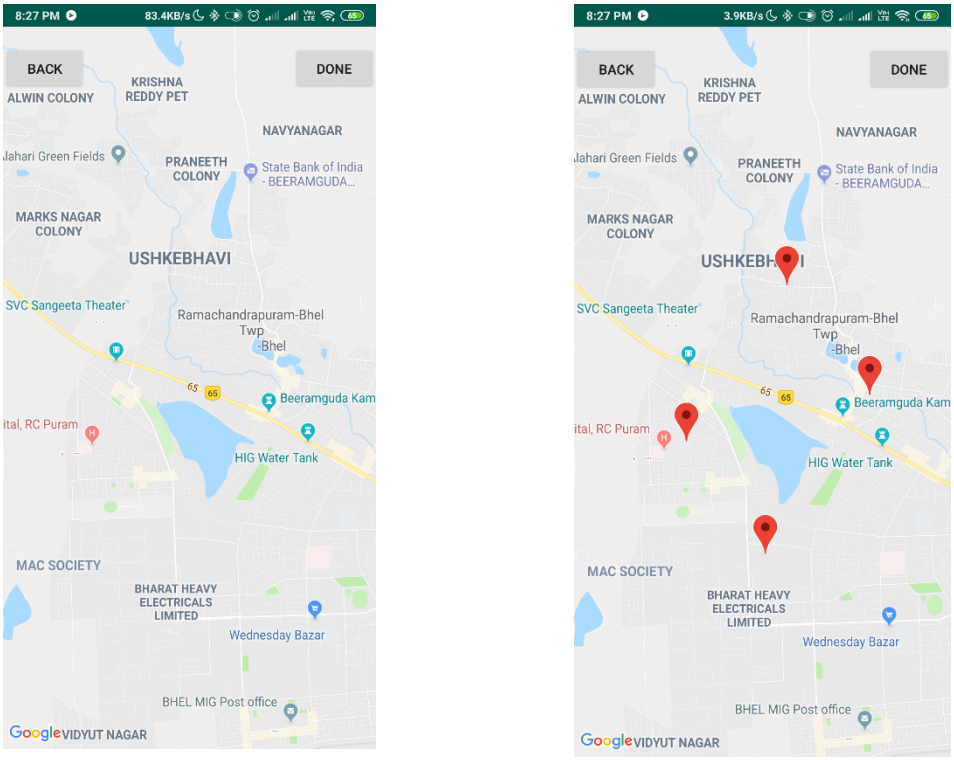


Fig-(6) : Selecting Locations on Maps

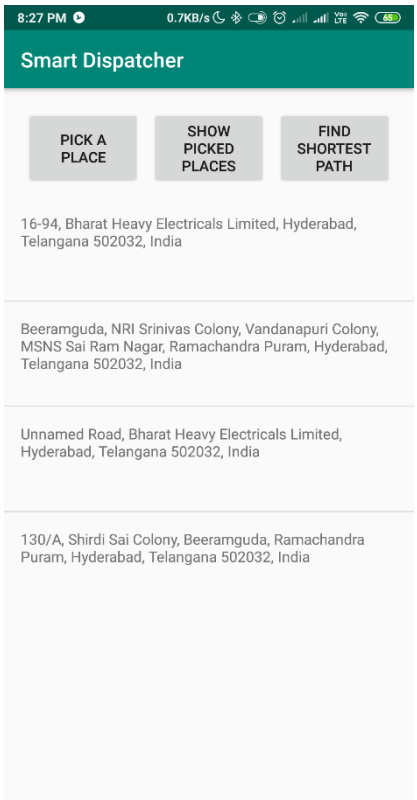


Fig-(7) : Displays Selected Locations

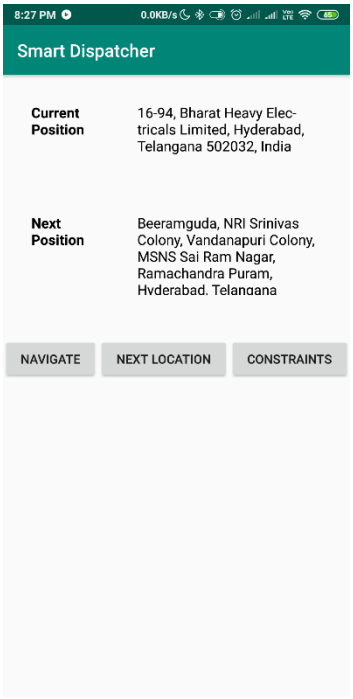


Fig-(8): Displays the Route

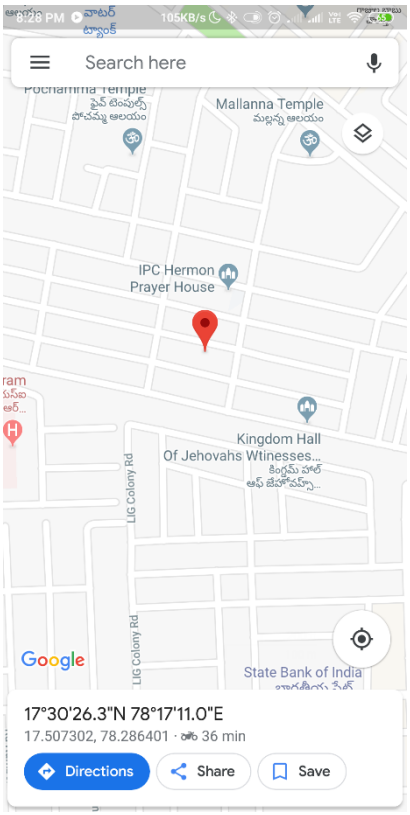
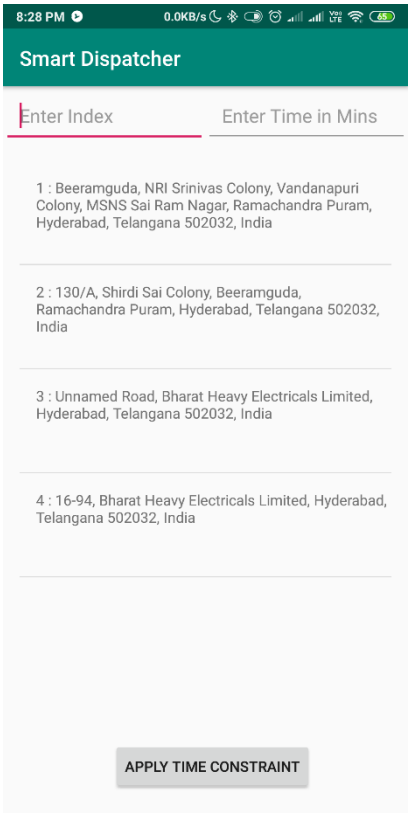


Fig-(9) :Navigation from Current location to next location



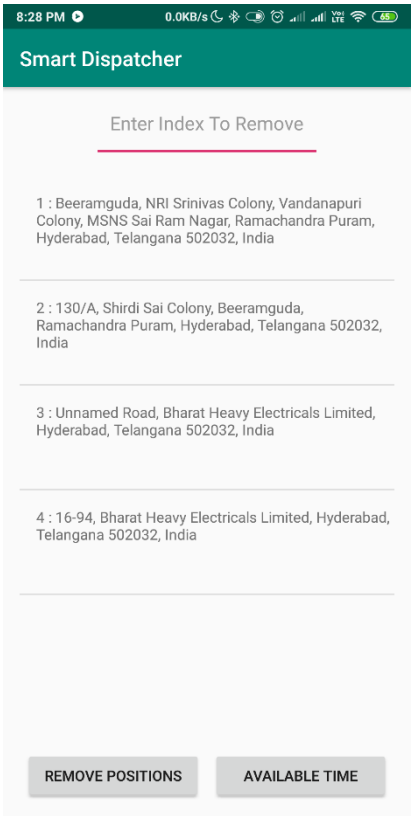


Fig-(10) : Adding Constraints on Locations

5. Conclusion and future work

The route provided considering the contraints provided and it is the best route which is of minimum distance .

Future Enhancements

The ‘distance’ function is used in the application to compute the distance matrix.The distances the function provides are not accurate.In order to overcome this Distance matrix api can be used to obtain distances accurately.

There may be other constraints such as traffic ,vehicle type which may be considered as a future enhancement.

References

[1] https://en.wikipedia.org/wiki/Shortest_path_problem

[2] https://developer.android.com/docs/

[3]https://github.com/williamfiset/Algorithms/blob/master/com/williamfiset/algorithms/graphtheory/TspDynamicProgrammingRecursive.java

[4] https://developers.google.com/maps/documentation/android-sdk/intro

[5] https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

Source Code

Travelling Salesman Algorithm code

```
import java.util.*;

public class TspDynamicProgrammingRecursive {

  private final int N;
  private final int START_NODE;
  private final int FINISHED_STATE;

  private double[][] distance;
  private double minTourCost = Double.POSITIVE_INFINITY;

  private List<Integer> tour = new ArrayList<>();
  private boolean ranSolver = false;

  public TspDynamicProgrammingRecursive(double[][] distance) {
    this(0, distance);
  }

  public TspDynamicProgrammingRecursive(int startNode, double[][] distance) {

    this.distance = distance;
    N = distance.length;
    START_NODE = startNode;
       FINISHED_STATE = (1 << N) - 1;
  }

  // Returns the optimal tour for the traveling salesman problem.
  public List<Integer> getTour() {
    if (!ranSolver) solve();
    return tour;
  }

  // Returns the minimal tour cost.
  public double getTourCost() {
    if (!ranSolver) solve();
    return minTourCost;
  }

  public void solve() {

    // Run the solver
    int state = 1 << START_NODE;
    Double[][] memo = new Double[N][1 << N];
    Integer[][] prev = new Integer[N][1 << N];
```

```
  minTourCost = tsp(START_NODE, state, memo, prev);

  // Regenerate path
  int index = START_NODE;
  while (true) {
    tour.add(index);
    Integer nextIndex = prev[index][state];
    if (nextIndex == null) break;
    int nextState = state | (1 << nextIndex);
    state = nextState;
    index = nextIndex;
  }
  tour.add(START_NODE);
  ranSolver = true;
}

private double tsp(int i, int state, Double[][] memo, Integer[][] prev) {

  // Done this tour. Return cost of going back to start node.
  if (state == FINISHED_STATE) return distance[i][START_NODE];

  // Return cached answer if already computed.
  if (memo[i][state] != null) return memo[i][state];

  double minCost = Double.POSITIVE_INFINITY;
  int index = -1;
  for (int next = 0; next < N; next++) {

    // Skip if the next node has already been visited.
    if ((state & (1 << next)) != 0) continue;

    int nextState = state | (1 << next);
    double newCost = distance[i][next] + tsp(next, nextState, memo, prev);
    if (newCost < minCost) {
      minCost = newCost;
      index = next;
    }
  }

  prev[i][state] = index;
  return memo[i][state] = minCost;
}

// Example usage:
public static void main(String[] args) {

  // Create adjacency matrix
  int n = 6;
  double[][] distanceMatrix = {{0,17,12,8,6,5,17,14,15,13},
```

```
            {17,0,8,24,27,22,32,52,23,20},

            {12,8,0,17,12,8,25,28,19,11},

            {8,24,17,0,10,5,18,23,12,12},

            {6,27,12,10,0,6,25,25,14,10},

            {5,22,8,5,6,0,38,29,23,8},

            {17,32,25,18,25,38,0,12,26,31},

            {14,52,28,23,25,29,12,0,33,35},

            {15,23,19,12,14,23,26,33,0,9},

            {13,20,11,12,10,8,31,35,9,0}};
    // Run the solver
    TspDynamicProgrammingRecursive solver = new
TspDynamicProgrammingRecursive(distanceMatrix);

    // Prints: [0, 3, 2, 4, 1, 5, 0]
            List<Integer> route = new ArrayList<>();
            route=solver.getTour();
    System.out.println("\nTour: " + route);
    System.out.println("Tour cost: " + solver.getTourCost());
  }

}
```

Java files

ConstraintAdapter.java

```
package com.example.android.smartdispatcher;

import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.TextView;

import java.util.ArrayList;
```

```
public class constraintadapter extends ArrayAdapter<locations> {

   int i = 0;

   public constraintadapter(Context context, ArrayList<locations> name) {
      super(context, 0, name);
   }

   public View getView(int position, View convertView, ViewGroup parent) {
      locations currentpositionwordobj = getItem(position);
      View listItemView = convertView;
      if (listItemView == null) {
         listItemView = LayoutInflater.from(getContext()).inflate(
               R.layout.checkboxshowlist, parent, false);
         i++;
      }
      Log.d("value",""+i);
      TextView miwokTextView = listItemView.findViewById(R.id.showtextview);
      miwokTextView.setText(currentpositionwordobj.getPosition()+" :
"+currentpositionwordobj.getAddress());
      View textContainer = listItemView.findViewById(R.id.checkboxlinear_id);
      return listItemView;

   }

}
```

Location.java

```
package com.example.android.smartdispatcher;

import com.google.android.gms.maps.model.LatLng;

public class locations {
   private LatLng latLng;
   private int position;
   private String address;
   public locations() {
   }
   public locations(String address) {
      this.address = address;
   }
   public LatLng getLatLng() {
      return latLng;
   }
   public void setLatLng(LatLng latLng) {
      this.latLng = latLng;
   }

   public String getAddress() {
```

```
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public int getPosition() {
        return position;
    }

    public void setPosition(int position) {
        this.position = position;
    }

}
```

Currentnextloc.java

```
package com.example.android.smartdispatcher;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import static android.location.Location.distanceBetween;
public class currentnextloc extends AppCompatActivity {
    ArrayList<String> latlist = new ArrayList<String>();
    ArrayList<String> addresslist = new ArrayList<String>();
    ArrayList<String> lonlist = new ArrayList<String>();
    String routepath = "";
    ArrayList<String> removelatlist = new ArrayList<String>();
    ArrayList<String> removeaddresslist = new ArrayList<String>();
    ArrayList<String> removelonlist = new ArrayList<String>();
    public String findlocation(ArrayList<String> removeaddresslist) {
        if (removeaddresslist.size() == 0) {
            return "no next location";
        }
        return removeaddresslist.get(0);
    }

    public String findcurrentlocation(ArrayList<String> removeaddresslist) {
```

```
    if (removeaddresslist.size() == 0) {
        return "no new location";
    }
    removelatlist.remove(0);
    removelonlist.remove(0);
    return removeaddresslist.remove(0);
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.currentnextloc);
    Bundle b = getIntent().getExtras();
    addresslist = b.getStringArrayList("addresslist");
    latlist = b.getStringArrayList("latlist");
    lonlist = b.getStringArrayList("lonlist");

    double[][] distanceMatrix = new double[addresslist.size()][addresslist.size()];
    for (int i = 0; i < addresslist.size(); i++) {
        for (int j = 0; j < addresslist.size(); j++) {
            float[] distance = new float[1];
            distanceBetween(Double.parseDouble(latlist.get(i)),
                    Double.parseDouble(lonlist.get(i)),
                    Double.parseDouble(latlist.get(j)),
                    Double.parseDouble(lonlist.get(j)),
                    distance);
            distanceMatrix[i][j] = distance[0] / 1000;
        }
    }
    tsp solver = new tsp(distanceMatrix);
    List<Integer> route = new ArrayList<>();
    route = solver.getTour();
    for (int i = 0; i < route.size(); i++) {
        routepath = routepath + route.get(i);
    }
    for (int i = 0; i < routepath.length(); i++) {
        removeaddresslist.add(addresslist.get(routepath.charAt(i) - 48));
        removelonlist.add(lonlist.get(routepath.charAt(i) - 48));
        removelatlist.add(latlist.get(routepath.charAt(i) - 48));
    }
    latlist = new ArrayList<String>();
    addresslist = new ArrayList<String>();
    lonlist = new ArrayList<String>();
    for (int i = 0; i < removelatlist.size(); i++) {
        latlist.add(removelatlist.get(i));
        addresslist.add(removeaddresslist.get(i));
        lonlist.add(removelonlist.get(i));
    }

    final TextView currentposition = findViewById(R.id.currentposition);
```

```
        TextView nextposition = findViewById(R.id.nextposition);

        currentposition.setText(findcurrentlocation(removeaddresslist));
        nextposition.setText(findlocation(removeaddresslist));

        Button constraintbutton = findViewById(R.id.constraintbutton);
        constraintbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (removeaddresslist.size() != 0) {
                    Intent intent = new Intent(currentnextloc.this, constraintlayout.class);
                    intent.putExtra("addresslist", removeaddresslist);
                    intent.putExtra("latlist", removelatlist);
                    intent.putExtra("lonlist", removelonlist);
                    startActivity(intent);
                } else {
                    Toast.makeText(currentnextloc.this, "no locations to apply constraint",
Toast.LENGTH_LONG).show();
                }

            }
        });

        Button nextbutton = findViewById(R.id.next);
        nextbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //show next location
                if (removeaddresslist.size() != 0) {
                    TextView currentposition = findViewById(R.id.currentposition);
                    TextView nextposition = findViewById(R.id.nextposition);
                    currentposition.setText(findcurrentlocation(removeaddresslist));
                    nextposition.setText(findlocation(removeaddresslist));
                }
            }
        });

        Button navigate = findViewById(R.id.navigate);
        navigate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int var = latlist.size() - removelatlist.size() - 1;
                Uri gmmIntentUri = Uri.parse("geo:0,0?q=" + latlist.get(var) + "," + lonlist.get(var)
+ "(" + addresslist.get(var) + ")");
                Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
                mapIntent.setPackage("com.google.android.apps.maps");
                if (mapIntent.resolveActivity(getPackageManager()) != null) {
                    startActivity(mapIntent);
                } else {
```

```
                Toast.makeText(currentnextloc.this, "Install Google Maps",
Toast.LENGTH_SHORT).show();
            }
        }
    });

  }
}
```

Constraintlayout

```
package com.example.android.smartdispatcher;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

public class constraintlayout extends AppCompatActivity {

  ArrayList<String> latlist;
  ArrayList<String> addresslist;
  ArrayList<String> lonlist;

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.constraintlayout);
    Bundle b = getIntent().getExtras();
    if (b == null) {
      Toast.makeText(constraintlayout.this, "select places first",
Toast.LENGTH_LONG).show();
    } else {
      addresslist = b.getStringArrayList("addresslist");
      latlist = b.getStringArrayList("latlist");
      lonlist = b.getStringArrayList("lonlist");
      ArrayList<locations> addresslocations = new ArrayList<locations>();
      for (int i = 0; i < addresslist.size(); i++) {
        locations dummy = new locations();
        dummy.setAddress(addresslist.get(i));
        dummy.setPosition((i + 1));
        addresslocations.add(dummy);
      }
```

```
        constraintadapter addressadapter = new constraintadapter(constraintlayout.this,
addresslocations);
        ListView listView = findViewById(R.id.checkboxlistview);
        listView.setAdapter(addressadapter);
      }
      Button removeposition = (Button) findViewById(R.id.removeposition);
      removeposition.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          int flag = 0;
          ArrayList<Integer> removeindex = new ArrayList<Integer>();
          EditText removeedittext = (EditText) findViewById(R.id.itemstoremove);
          String removinglist = removeedittext.getText().toString();
          String numbers[] = removinglist.split(",");
          for (int i = 0; i < numbers.length; i++) {
            if (Integer.parseInt(numbers[i]) <= (addresslist.size() + 1)) {
              removeindex.add(Integer.parseInt(numbers[i]));
            } else {
              removeindex = new ArrayList<Integer>();
              flag = 1;
              Toast.makeText(constraintlayout.this, "enter correct index",
Toast.LENGTH_LONG).show();
            }
          }
          if (flag == 0) {
//            Intent intent = new Intent(constraintlayout.this, currentnextloc.class);
//            intent.putExtra("removingindexlist", removeindex);
//            startActivity(intent);
          }
        }
      });
    }
}
```

Login.java

```
package com.example.android.smartdispatcher;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class login extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);


        Button loginbutton= findViewById(R.id.loginbutton);
        loginbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String crrtmobileno="9492914041";
                String crrtpassword="12345678";

                EditText mobileno= findViewById(R.id.mobileno);
                EditText password= findViewById(R.id.password);

                if(mobileno.getText().toString().equals(crrtmobileno) &&
password.getText().toString().equals(crrtpassword))
                {
                    Intent intent=new Intent(login.this,MainActivity.class);
                    startActivity(intent);
                }
                else
                {
                    Toast.makeText(login.this,"Enter Valid
Credentials",Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

MainActivity.java

```
package com.example.android.smartdispatcher;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {

    ArrayList<String> latlist;
    ArrayList<String> addresslist;
    ArrayList<String> lonlist;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final Button placepickerbutton = findViewById(R.id.pickplace);
        placepickerbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, MapsActivity.class);
                startActivity(intent);
            }
        });

        final Button findpathButton = findViewById(R.id.findpath);
        findpathButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle b = getIntent().getExtras();
                if (b == null) {
                    Toast.makeText(MainActivity.this, "select places first",
Toast.LENGTH_LONG).show();
                } else {
                    addresslist = b.getStringArrayList("addresslist");
                    latlist = b.getStringArrayList("latlist");
                    lonlist = b.getStringArrayList("lonlist");
                    if (addresslist.size() <= 2) {
                        Toast.makeText(MainActivity.this, "select more than 2 places",
Toast.LENGTH_LONG).show();
                    } else {
                        Intent intent = new Intent(MainActivity.this, currentnextloc.class);
                        intent.putExtra("addresslist", addresslist);
                        intent.putExtra("latlist", latlist);
                        intent.putExtra("lonlist", lonlist);
                        startActivity(intent);
                    }
                }
            }
        });
        final Button showplacesButton = findViewById(R.id.showplaces);
        showplacesButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bundle b = getIntent().getExtras();
                if (b == null) {
                    Toast.makeText(MainActivity.this, "select places first",
Toast.LENGTH_LONG).show();
                } else {
                    addresslist = b.getStringArrayList("addresslist");
                    latlist = b.getStringArrayList("latlist");
                    lonlist = b.getStringArrayList("lonlist");
                    ArrayList<locations> addresslocations = new ArrayList<locations>();
                    for (int i = 0; i < addresslist.size(); i++) {
```

```
                addresslocations.add(new locations(addresslist.get(i)));
            }
            Addressadapter addressadapter = new Addressadapter(MainActivity.this,
addresslocations);
            ListView listView = findViewById(R.id.listview);
            listView.setAdapter(addressadapter);
        }
      }
    });

  }
}
```

MapsActivity.java

```
package com.example.android.smartdispatcher;

import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.View;
import android.widget.Button;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

  ArrayList<String> addresslist = new ArrayList<String>();
  ArrayList<String> latlist=new ArrayList<String>();
  ArrayList<String> lonlist=new ArrayList<String>();
  private GoogleMap mMap;
  private Button back;
  private Button done;
  private LatLng latlong;
  private String s = "";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
```

```
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        back = findViewById(R.id.backButton);
        done = findViewById(R.id.doneButton);

        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MapsActivity.this, MainActivity.class);
                startActivity(intent);
            }
        });
        done.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MapsActivity.this, MainActivity.class);
                intent.putExtra("addresslist",addresslist);
                intent.putExtra("latlist",latlist);
                intent.putExtra("lonlist",lonlist);
                startActivity(intent);
            }
        });

    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        LatLng bhel = new LatLng(17.5116485, 78.2932053);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(bhel, 14));
        mMap.setOnMapLongClickListener(new GoogleMap.OnMapLongClickListener() {
            @Override
            public void onMapLongClick(LatLng latLng) {
                latlong = latLng;
                mMap.addMarker(new MarkerOptions()
                        .position(latLng)
                        .title(getAddress(latLng.latitude, latLng.longitude)));
                addresslist.add(getAddress(latLng.latitude, latLng.longitude));
                latlist.add(String.valueOf(latLng.latitude));
                lonlist.add(String.valueOf(latLng.longitude));
            }
        });
    }

    public String getAddress(double lat, double lng) {
        Geocoder geocoder = new Geocoder(MapsActivity.this, Locale.getDefault());
        try {
            List<Address> addresses = geocoder.getFromLocation(lat, lng, 1);
```

```
            Address obj = addresses.get(0);
            return obj.getAddressLine(0);

        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Removetsp.java

```
package com.example.android.smartdispatcher;

import java.util.ArrayList;
import java.util.List;

public class removetsp  {
    private final int N;
    private final int START_NODE;
    private final int FINISHED_STATE;

    private double[][] distance;
    private double minTourCost = Double.POSITIVE_INFINITY;

    private List<Integer> tour = new ArrayList<>();
    private boolean ranSolver = false;

    public removetsp(double[][] distance) {
        this(0, distance);
    }

    public removetsp(int startNode, double[][] distance) {

        this.distance = distance;
        N = distance.length;
        START_NODE = startNode;
        FINISHED_STATE = (1 << N) - 1;
    }

    public List<Integer> getTour() {
        if (!ranSolver) solve();
        return tour;
    }

    public double getTourCost() {
        if (!ranSolver) solve();
        return minTourCost;
    }
```

```
    public void solve() {

        int state = 1 << START_NODE;
        Double[][] memo = new Double[N][1 << N];
        Integer[][] prev = new Integer[N][1 << N];
        minTourCost = tsp(START_NODE, state, memo, prev);

        int index = START_NODE;
        while (true) {
            tour.add(index);
            Integer nextIndex = prev[index][state];
            if (nextIndex == null) break;
            int nextState = state | (1 << nextIndex);
            state = nextState;
            index = nextIndex;
        }
        ranSolver = true;
    }

    private double tsp(int i, int state, Double[][] memo, Integer[][] prev) {

        if (state == FINISHED_STATE) return 0;

        if (memo[i][state] != null) return memo[i][state];

        double minCost = Double.POSITIVE_INFINITY;
        int index = -1;
        for (int next = 0; next < N; next++) {

            if ((state & (1 << next)) != 0) continue;

            int nextState = state | (1 << next);
            double newCost = distance[i][next] + tsp(next, nextState, memo, prev);
            if (newCost < minCost) {
                minCost = newCost;
                index = next;
            }
        }
        prev[i][state] = index;
        return memo[i][state] = minCost;
    }
}
```

Tsp.java

```
package com.example.android.smartdispatcher;

import java.util.ArrayList;
import java.util.List;
```

```
public class tsp {
   private  int N;
   private  int START_NODE;
   private  int FINISHED_STATE;

   private double[][] distance;
   private double minTourCost = Double.POSITIVE_INFINITY;

   private List<Integer> tour = new ArrayList<>();
   private boolean ranSolver = false;

   public tsp() {

   }

   public tsp(double[][] distance) {
      this(0, distance);
   }

   public tsp(int startNode, double[][] distance) {

      this.distance = distance;
      N = distance.length;
      START_NODE = startNode;
      FINISHED_STATE = (1 << N) - 1;
   }

   public List<Integer> getTour() {
      if (!ranSolver) solve();
      return tour;
   }

   public void solve() {

      // Run the solver
      int state = 1 << START_NODE;
      Double[][] memo = new Double[N][1 << N];
      Integer[][] prev = new Integer[N][1 << N];
      minTourCost = tsp(START_NODE, state, memo, prev);

      int index = START_NODE;
      while (true) {
         tour.add(index);
         Integer nextIndex = prev[index][state];
         if (nextIndex == null) break;
         int nextState = state | (1 << nextIndex);
         state = nextState;
         index = nextIndex;
      }
```

```
        tour.add(START_NODE);
        ranSolver = true;
    }

    private double tsp(int i, int state, Double[][] memo, Integer[][] prev) {

        if (state == FINISHED_STATE) return distance[i][START_NODE];

        if (memo[i][state] != null) return memo[i][state];

        double minCost = Double.POSITIVE_INFINITY;
        int index = -1;
        for (int next = 0; next < N; next++) {

            if ((state & (1 << next)) != 0) continue;

            int nextState = state | (1 << next);
            double newCost = distance[i][next] + tsp(next, nextState, memo, prev);
            if (newCost < minCost) {
                minCost = newCost;
                index = next;
            }
        }
        prev[i][state] = index;
        return memo[i][state] = minCost;
    }
}
```

Layouts

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:paddingLeft="16dp"
```

```
        android:paddingTop="16dp"
        android:paddingRight="16dp"
        android:paddingBottom="5dp">

        <Button
            android:id="@+id/pickplace"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="pick a place" />

        <Button
            android:id="@+id/showplaces"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="show picked places" />

        <Button
            android:id="@+id/findpath"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="find shortest path" />

    </LinearLayout>

    <ListView
        android:id="@+id/listview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    </ListView>

  </LinearLayout>


</android.support.constraint.ConstraintLayout>
```

Activity_maps.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" >


    <Button
        android:layout_width="wrap_content"
        android:id="@+id/doneButton"
        android:layout_height="wrap_content"
        android:layout_gravity="right|top"
        android:text="Done"
        android:padding="10dp"
        android:layout_marginTop="20dp"
        android:paddingRight="10dp"/>
    <Button
        android:id="@+id/backButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top"
        android:text="Back"
        android:padding="10dp"
        android:layout_marginTop="20dp"
        android:paddingRight="10dp"/>

</fragment>
```

Addresshowlist.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/linear_id"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/addressview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="16dp"
            android:text="address" />
```

```
    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Checkboxshowlist.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/checkboxlinear_id"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/showtextview"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="16dp"
            android:text="address" />
    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Constraintlayout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <EditText
            android:id="@+id/itemstoremove"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:hint="Enter Index To Remove"
            android:padding="16dp" />
```

```
        <ListView
            android:id="@+id/checkboxlistview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="8"
            android:padding="16dp">

        </ListView>


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal"
            android:paddingLeft="16dp"

            android:paddingTop="16dp"
            android:paddingRight="16dp"
            android:paddingBottom="5dp">

            <Button
                android:id="@+id/removeposition"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_margin="5dp"
                android:layout_weight="1"
                android:text="remove positions" />

            <Button
                android:id="@+id/timeconstraint"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_margin="5dp"
                android:layout_weight="1"
                android:elevation="0dp"
                android:text="available time" />

        </LinearLayout>

    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Currentnextloc.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <ScrollView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:layout_editor_absoluteX="0dp"
      tools:layout_editor_absoluteY="0dp">

      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:orientation="vertical">

         <LinearLayout

            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:orientation="horizontal">

            <LinearLayout
               android:layout_width="0dp"
               android:layout_height="250dp"
               android:layout_weight="1"
               android:orientation="vertical">

               <TextView
                  android:layout_width="wrap_content"
                  android:layout_height="125dp"
                  android:layout_gravity="center"
                  android:layout_weight="1"
                  android:padding="16dp"
                  android:text="Current Position"
                  android:textColor="@android:color/black"
                  android:textSize="16sp"
                  android:textStyle="bold" />

               <TextView
                  android:layout_width="wrap_content"
                  android:layout_height="125dp"
                  android:layout_gravity="center"
                  android:layout_weight="1"
                  android:padding="16dp"
                  android:text="Next Position"
                  android:textColor="@android:color/black"
```

```
            android:textSize="16sp"
            android:textStyle="bold" />


    </LinearLayout>

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="250dp"
        android:layout_weight="2"
        android:orientation="vertical">

        <TextView
            android:id="@+id/currentposition"
            android:layout_width="wrap_content"
            android:layout_height="125dp"
            android:layout_gravity="center"
            android:padding="16dp"
            android:text="hello"
            android:textColor="@android:color/black"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/nextposition"
            android:layout_width="wrap_content"
            android:layout_height="125dp"
            android:layout_gravity="center"
            android:padding="16dp"
            android:text="hello2"
            android:textColor="@android:color/black"
            android:textSize="16sp" />
    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/navigate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="navigate" />

    <Button
        android:id="@+id/next"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="next location" />

        <Button
            android:id="@+id/constraintbutton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="constraints" />
    </LinearLayout>

    <ListView
        android:id="@+id/unvisitedlist"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

  </LinearLayout>
 </ScrollView>
</android.support.constraint.ConstraintLayout>
```

Login.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <ScrollView
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:layout_gravity="center_vertical"
         android:gravity="center_horizontal"
         android:orientation="vertical">

         <EditText
            android:id="@+id/mobileno"
            android:layout_width="250dp"
            android:layout_height="wrap_content"
            android:hint="Mobile Number"
            android:inputType="phone"
            android:textSize="16sp" />
```

```
        <EditText
            android:id="@+id/password"
            android:layout_width="250dp"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:inputType="numberPassword"
            android:padding="16dp"
            android:textSize="16sp" />

        <Button
            android:id="@+id/loginbutton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:padding="16dp"
            android:text="login" />
    </LinearLayout>
  </ScrollView>
</android.support.constraint.ConstraintLayout>
```

Timelayout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:layout_editor_absoluteX="153dp"
    tools:layout_editor_absoluteY="0dp">

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal">

      <EditText
        android:id="@+id/indexno"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:hint="Enter Index "
        android:padding="16dp" />
```

```
        <EditText
            android:id="@+id/time"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:hint="Enter Time in Mins"
            android:padding="16dp" />
    </LinearLayout>

    <ListView
        android:id="@+id/timelistview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="8"
        android:padding="16dp">

    </ListView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:paddingLeft="16dp"
        android:paddingTop="16dp"
        android:paddingRight="16dp"
        android:paddingBottom="5dp">

        <Button
            android:id="@+id/timeconstraintbutton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="90dp"
            android:text="Apply Time Constraint" />

    </LinearLayout>

  </LinearLayout>
</android.support.constraint.ConstraintLayout>
```