MIT Academy of Engineering, Alandi (D)

Department of Computer Engineering

INDEX

Sr No	Name of Experiment	Date	Page No	Sign
Group A				
1	Implementation of any 2 uninformed search methods with some application			
2	Write a program to perform profile translation-based proactive adaptation using context management in smartphones. Objective of this assignment is to automatically generates user's profile according to the scenarios using machine learning approaches. System should allow to keep user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information			
3	Implement A* approach for any suitable application			
4	Implementation of Unification algorithm			
5	Implement Naive Bayes to predict the work type for a person ,			
6	Design an application for Pedometer related to Android OS.			
Group B				
1	Write a program to build smart mobile app for user profiling. Objective of this assignment is to develop smart mobile app which can create user profiles based on their preferences so that smart recommendations can be provided at run time.			
2	Implementation of MiniMax approach for TIC-TAC-TOE using Java/ Scala/ Python-Eclipse Use GUI Player X and Player O are using their mobiles for the play. Refresh the screen after the move for both the players.			
3	Implementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using XML/JSON use C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities			

4	Developing an book recommender (a book that the reader should read and is new			
5	Implement k-means for clustering data of children belonging to different age groups to perform some specific activities. Formulate the Feature vector for following parameters: i. Height ii. Weight iii. Age iv. IQ Formulate the data for 40 children to form 3 clusters.			
6	Write a mobile app using Scala/ Python/ C++/ Android using Eclipse to beep the mobile speaker for three incorrect attempts of the password.			
Group C				
1	Write a program to Smart Watch App Development with Tizen. Objective of this assignment is to design simple comic app with the Tizen SDK for Wearable and run it on the smartwatch emulator that comes bundled with the IDE.			

Subject Teacher   Head of Dept.

Group A

Assignment No 1

Aim: Implementation of any 2 uninformed search methods with some application.

Objectives: To understand the uninformed search for space search and efficient routing of graphs.

Tools Requirement:

Hardware: - 500 MB Hard Disk, 512 MB RAM.

Software: - JDK 6 or above, gcc-c++, Python 2.x

Theory:

Uninformed Search

The uninformed search methods offer a variety of techniques for graphsearch, each with its own advantages and disadvantages. These methods areexplored here with a discussion of their characteristics and complexities.

Big-O notation will be used to compare the algorithms. This notationdefines the asymptotic upper bound of the algorithm given the depth (*d*) ofthe tree and the branching factor, or the average number of branches (*b*)from each node. There are a number of common complexities that exist forsearch algorithms.A search algorithm is characterized as exhaustive when it can search every node in the graph in search of the goal.If the goal is not present in the graph, the algorithm will terminate, but will search each and every node in a systematic way.

Common orders of search functions.

O-Notation	Order
O(1)	Constant (regardless of the number of nodes)
O(n)	Linear (consistent with the number of nodes)
O(log n)	Logarithmic
$O(n^2)$	Quadratic
$O(c^n)$	Geometric
O(n!)	Combinatorial

Big-O notation provides a worst-case measure of the complexity of a search algorithm and is a common comparison tool for algorithms. We'll compare the search algorithms using *space complexity* (a measure of the memory required during the search) and *time complexity* (worst-case time required to find a solution). We'll also review the algorithm for *completeness* (can the algorithm find a path to a goal node if it's present in the graph) and *optimality* (finds the lowest cost solution available).

Depth-First Search (DFS)

The Depth-First Search (DFS) algorithm is a technique for searching a graph that begins at the root node, and exhaustively searches each branch to its greatest depth before backtracking to previously unexplored branches. Nodes found but yet to be reviewed are stored in a LIFO queue (also known as a *stack*).

The space complexity for DFS is $O(b^d)$ where the time complexity is geometric ($O(b^d)$). This can be

very problematic on deep branching graphs, as the algorithm will continue to the maximum depth of the graph. If loops are present in the graph, then DFS will follow these cycles indefinitely. For this reason, the DFS algorithm is not complete, as cycles can prohibit the algorithm from finding the goal. If cycles are not present in the graph,then the algorithm is complete (will always find the goal node). The DFS algorithm is also not optimal, but can be made optimal using path checking (to ensure the shortest path to the goal is found).
Graph algorithms can be implemented either recursively or using a stack to maintain the list of nodes that must be enumerated.

Depth-Limited Search (DLS)

Depth-Limited Search (DLS) is a modification of depth-first search that minimizes the depth that the search algorithm may go. In addition to starting with a root and goal node, a depth is provided that the algorithm will notdescend below. Any nodes below that depth are omitted fromthe search. This modification keeps the algorithm from indefinitely cyclingby halting the search after the pre-imposed depth.

While the algorithm does remove the possibility of infinitely looping in thegraph, it also reduces the scope of the search. If the goal node had been oneof the nodes marked 'X', it would not have been found, making the searchalgorithm incomplete. The algorithm can be complete if the search depth isthat of the tree itself (in this case *d* is three). The technique is also not optimalsince the first path may be found to the goal instead of the shortest path.The time and space complexity of depth-limited search is similar to DFS,from which this algorithm is derived. Space complexity is O(b^d) and timecomplexity is O(b^d), but *d* in this case is the imposed depth of the search andnot the maximum depth of the graph.

Iterative Deepening Search (IDS)

Iterative Deepening Search (IDS) is a derivative of DLS and combines the features of depth-first search with that of breadth-first search. IDS operatesby performing DLS searches with increased depths until the goal is found.The depth begins at one, and increases until the goal is found, or no furthernodes can be enumerated.

IDS combines depth-first search with breadthfirst search. By minimizing the depth of the search, we force the algorithm toalso search the breadth of the graph. If the goal is not found, the depth thatthe algorithm is permitted to search is increased and the algorithm is startedagain. IDS is advantageous because it's not susceptible to cycles (a characteristicof DLS, upon which it's based). It also finds the goal nearest to the root node,as does the BFS algorithm (which will be detailed next). For this reason, it'sa preferred algorithm when the depth of the solution is not known.The time complexity for IDS is identical to that of DFS and DLS, O(b^d).
Space complexity of IDS is O(b^d).Unlike DFS and DLS, IDS is will always find the best solution and therefore, it is both complete and optimal.

Breadth-First Search (BFS)

In Breadth-First Search (BFS), we search the graph from the root node inorder of the distance from the root. Because the order search is nearest the root, BFS is guaranteed to find the best possible solution (shallowest) in anon-weighted graph, and is therefore also complete. Rather than digging deep down into the graph, progressing further and further from the root (as is the case with DFS), BFS checks each

node nearest the root beforedescending to the next level.
The implementation of BFS uses a FIFO (first-in-first-out) queue, differing from the stack (LIFO) implementation for DFS. As new nodes are found to be searched, these nodes are checked against the goal, and ifthe goal is not found, the new nodes are added to the queue. To continue the search, the oldest node is dequeued (FIFO order). Using FIFO orderfor new node search, we always check the oldest nodes first, resulting inbreadth-first review.

The disadvantage of BFS is that each node that is searched is requiredto be stored (space complexity is O(b^d)). The entire depth of the tree doesnot have to be searched, so *d* in this context is the depth of the solution, and
not the maximum depth of the tree. Time complexity is also O(b^d).
In practical implementations of BFS, and other search algorithms, aclosed list is maintained that contains those nodes in the graph thathave been visited. This allows the algorithm to efficiently search thegraph without re-visiting nodes. In implementations where the graph isweighted, keeping a closed list is not possible.

Bidirectional Search
The Bidirectional Search algorithm is a derivative of BFS that operates byperforming two breadth-first searches simultaneously, one beginning fromthe root node and the other from the goal node. When the two searchesmeet in the middle, a path can be reconstructed from the root to the goal.The searches meeting is determined when a common node is found. This is accomplished by keepinga closed list of the nodes visited.
Bidirectional search is an interesting idea, but requires that we know thegoal that we're seeking in the graph. This isn't always practical, which limitsthe application of the algorithm. When it can be determined, the algorithmhas useful characteristics. The time and space complexity for bidirectionalsearch is O(bd/2), since we're only required to search half of the depth ofthe tree. Since it is based on BFS, bidirectional search is both complete andoptimal.

Uniform-Cost Search (UCS)
One advantage of BFS is that it always finds the shallowest solution. Butconsider the edge having a cost associated with it. The shallowest solutionmay not be the best, and a deeper solution with a reduced path cost wouldbe better. Uniform -Cost Search (UCS) can be applied to find the least-cost path through a graph by maintaining anordered list of nodes in order of descending cost. This allows us to evaluatethe least cost path first
Uniform-cost search is an uninformed search method because no heuristicis actually used. The algorithm measures the actual cost of the pathwithout attempting to estimate it.

The algorithm for UCS uses the accumulated path cost and a priority queue to determine the path to evaluate. The priority queue(sorted from least cost to greatest) contains the nodes to be evaluated. Asnode children are evaluated, we add their cost to the node with the aggregatesum of the current path. This node is then added to the queue, and whenall children have been evaluated, the queue is sorted in order of ascendingcost. When the first element in the priority queue is the goal node, then thebest solution has been found.

The UCS algorithm is easily demonstrated using our example graph shows the state of the priority queue as the nodesare evaluated. At step one, the initial node has been added to the priorityqueue, with a cost of zero. At step two, each of the three connected nodesare evaluated and added to the priority queue. When no further children areavailable to evaluate, the priority queue is sorted to place them in ascendingcost order.

UCS is optimal and can be complete, but only if the edge costs arenon-negative (the summed path cost always increases). Time and spacecomplexity are the same as BFS, O(bd) for each, as it’s possible for the entiretree to be evaluated.

Program with proper indentation& comments:

//Depth First Search

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
 int i;
 reach[v]=1;
 for(i=1;i<=n;i++)
  if(a[v][i] && !reach[i])
  {
   printf("\n %d->%d",v,i);
   dfs(i);
  }
}
void main()
{
 int i,j,count=0;
 clrscr();
 printf("\n Enter number of vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  reach[i]=0;
  for(j=1;j<=n;j++)
   a[i][j]=0;
 }
 printf("\n Enter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
   scanf("%d",&a[i][j]);
 dfs(1);
 printf("\n");
 for(i=1;i<=n;i++)
 {
  if(reach[i])
   count++;
 }
 if(count==n)
  printf("\n Graph is connected");
 else
```

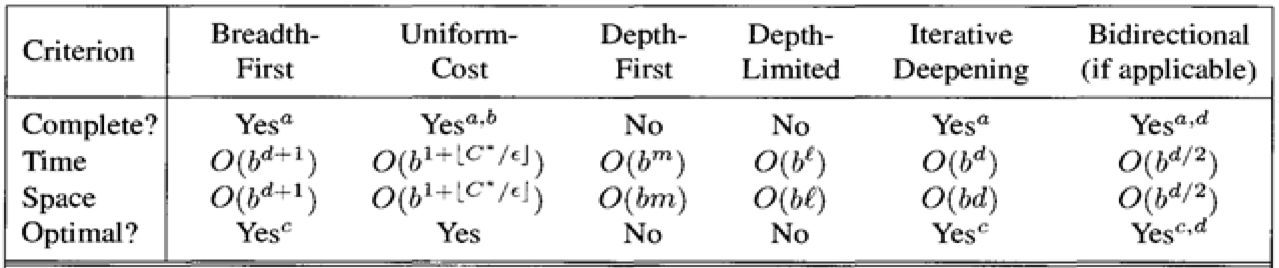
```
  printf("\n Graph is not connected");
 getch();
//Depth Limited Search
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.Stack;

public class DepthLimitedSearch {

    private Stack<Integer> stack;
    private int numberOfNodes;
    private static final int MAX_DEPTH = 3;

    public DepthLimitedSearch(int numberOfNodes) {
        this.numberOfNodes = numberOfNodes;
        this.stack = new Stack<Integer>();
    }

    public void depthLimitedSearch(int adjacencyMatrix[][], int source) {
        int visited[] = new int[numberOfNodes + 1];
        int element, destination;
        int depth = 0;

        System.out.println(source + " at depth " + depth);
        stack.push(source);
        visited[source] = 1;
        depth = 0;

        while (!stack.isEmpty()) {
            element = stack.peek();
            destination = element;
            while (destination <= numberOfNodes) {
                if (depth < MAX_DEPTH) {
                    if (adjacencyMatrix[element][destination] == 1 && visited[destination] == 0) {
                        stack.push(destination);
                        visited[destination] = 1;
                        depth++;
                        System.out.println(destination + " at depth " + depth);
                        element = destination;
                        destination = 1;
                    }
                } else {
                    return;
                }
                destination++;
            }
            stack.pop();
            depth--;
        }
    }

    public static void main(String[] args) {
        int number_of_nodes, source;
        Scanner scanner = null;
        try {
            System.out.println("Enter the number of nodes in the graph");
            scanner = new Scanner(System.in);
            number_of_nodes = scanner.nextInt();

            int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
            System.out.println("Enter the adjacency matrix");
            for (int i = 1; i <= number_of_nodes; i++) {
                for (int j = 1; j <= number_of_nodes; j++) {
                    adjacency_matrix[i][j] = scanner.nextInt();
                }
            }

            System.out.println("Enter the source for the graph");
            source = scanner.nextInt();

            System.out.println("The Depth limited Search Traversal of Max Depth 3 is");
            DepthLimitedSearch depthLimitedSearch = new DepthLimitedSearch(number_of_nodes);
            depthLimitedSearch.depthLimitedSearch(adjacency_matrix, source);

        } catch (InputMismatchException inputMismatch) {
            System.out.println("Wrong Input format");
        }
        scanner.close();
    }
}
```

Output:

```
$javac DepthLimitedSearch.java
$java DepthLimitedSearch
Enter the number of nodes in the graph
5
Enter the adjacency matrix
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0
```

```
Enter the source for the graph
1
The Depth limited Search Traversal  of Max Depth 3 for the graph is given by
1 at depth 0
2 at depth 1
3 at depth 2
4 at depth 3
```

Code complexity:



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.17 Evaluation of search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Conclusions: In this way, we have learnt about uniform cost search for the search space problems in artificial intelligence.

Assignment No 2

Aim: Reminder system for translation based proactive adaption using context management.

Tools:
Software: Java, C++, C, Python.
Hardware: Android OS based phones.

Theory:

Nowadays, mobile phones are used everywhere and every time in our daily life ranging from communication, entertainment to health and wellness applications. For managing incontinence among dementia subjects at workplace, it is difficult to have timely change with regular scheduled checks. Delays in change results serious implications in economic, social and clinical aspects. To enable timely workplace change by sensing its need and prompting for assistance. Besides this, a simple and easy to use reminder is important to attract immediate attentions from care givers. With trials conducted at nursing home, real user needs and limitations of initial reminder system can be identified. Based on application requirements and user needs, smart phone reminder system is designed and integrated into iCMS to effectively notify the care givers for timely change; thus, enhancing quality of incontinence management practices at workplace.

Program:

```
package com.appsrox.remindme;

import java.util.Calendar;
import java.util.Date;

import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.SimpleCursorAdapter;
import android.widget.SimpleCursorAdapter.ViewBinder;
import android.widget.TextView;
import android.widget.Toast;

import com.appsrox.remindme.model.Alarm;
import com.appsrox.remindme.model.AlarmMsg;
public class MainActivity extends ListActivity {

        private SQLiteDatabase database;
        private AlertDialog alertDialog;

        public final Calendar cal = Calendar.getInstance();
        public final Date mDate = new Date();
        private String[] monthArr;
        private AlarmMsg alarmMsg = new AlarmMsg();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        database = RemindMe.database;
        monthArr = getResources().getStringArray(R.array.spinner3_arr);

        //register context menu
                registerForContextMenu(getListView());

    }



        @SuppressWarnings("deprecation")
        private Cursor createCursor() {
                Cursor c = RemindMe.dbHelper.getListNotifications(database);
                startManagingCursor(c);
                return c;
        }

    @Override
        protected void onResume() {
                super.onResume();

                @SuppressWarnings("deprecation")
                SimpleCursorAdapter adapter = new SimpleCursorAdapter(
                                this,
                                R.layout.row,
                                createCursor(),
                                new String[]{Alarm.COL_NAME, AlarmMsg.COL_DATETIME,
AlarmMsg.COL_DATETIME, AlarmMsg.COL_DATETIME, AlarmMsg.COL_DATETIME},
                                new int[]{R.id.msg_tv, R.id.year_tv, R.id.month_tv, R.id.date_tv,
R.id.time_tv});

                adapter.setViewBinder(new ViewBinder() {
                    @SuppressWarnings("deprecation")
                    @Override
                    public boolean setViewValue(View view, Cursor cursor, int columnIndex) {
                                if (view.getId() == R.id.msg_tv)
                                {
                                return false;
                                }
                                TextView tv = (TextView)view;
                                long time = cursor.getLong(columnIndex);
                                mDate.setTime(time);
                                switch(view.getId()) {
                                case R.id.year_tv:
                                tv.setText(String.valueOf(mDate.getYear() + 1900));
                                break;
                                case R.id.month_tv:
                                tv.setText(monthArr[mDate.getMonth()+1]);
                                break;
                                case R.id.date_tv:
                                tv.setText(String.valueOf(mDate.getDate()));
                                break;
                                case R.id.time_tv:
                                long now = System.currentTimeMillis();
                                String txt = RemindMe.showRemainingTime() ?
Util.getRemainingTime(time, now) : Util.getActualTime(mDate.getHours(), mDate.getMinutes());
                                if (TextUtils.isEmpty(txt)) txt =
Util.getActualTime(mDate.getHours(), mDate.getMinutes());
                                tv.setText(txt);

                                RelativeLayout parent = (RelativeLayout)tv.getParent();
                                TextView tv2 = (TextView) parent.findViewById(R.id.msg_tv);
                                if (time < now) tv2.setTextColor(Color.parseColor("#555555"));
                                else tv2.setTextColor(Color.parseColor("#587498"));
                                break;
                                }
                                return true;
                    }
                });
                setListAdapter(adapter);

        }
        public void onClick(View v) {
        startActivity(new Intent(this, AddAlarmActivity.class));
        }

        @Override
```

```
	public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
		if (v.getId() == android.R.id.list) {
			getMenuInflater().inflate(R.menu.context_menu, menu);
			menu.setHeaderTitle("Choose an Option");
			menu.setHeaderIcon(R.drawable.ic_dialog_menu_generic);

			AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) menuInfo;
			alarmMsg.setId(info.id);
			alarmMsg.load(database);
			if (alarmMsg.getDateTime() < System.currentTimeMillis())
				menu.removeItem(R.id.menu_edit);
		}
	}

	@Override
	public boolean onContextItemSelected(MenuItem item) {
		AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();

			RemindMe.dbHelper.cancelNotification(database, info.id, false);

			Intent cancelThis = new Intent(this, AlarmService.class);
			cancelThis.putExtra(AlarmMsg.COL_ID, String.valueOf(info.id));
			cancelThis.setAction(AlarmService.CANCEL);
			startService(cancelThis);

			//Refresh adapter
			SimpleCursorAdapter adapter = (SimpleCursorAdapter) getListAdapter();
		adapter.getCursor().requery();
		adapter.notifyDataSetChanged();

		return true;
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		openContextMenu(v);
	}


	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		getMenuInflater().inflate(R.menu.options_menu, menu);
        return true;
	}

	@Override
	public boolean onPrepareOptionsMenu(Menu menu) {
    if (getListAdapter().isEmpty()) {
       menu.findItem(R.id.menu_delete_all).setEnabled(false);
    } else {
       menu.findItem(R.id.menu_delete_all).setEnabled(true);
    }
                return true;
       }



       @Override
       protected void onDestroy() {
                if (alertDialog != null)
                         alertDialog.dismiss();
                super.onDestroy();
       }

}
```

Conclusions: In this way we have studied the recommender system for context management.

Assignment No 3

Aim: Implementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using XML/JSON use C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities.

Objectives: To understand the uninformed search for space search and implement it in real world scenario.

Tools Requirement:

Hardware: - 500 MB Hard Disk, 512 MB RAM.

Software: - JDK 6 or above, gcc-c++, Python 2.x

Theory:

<u>Algorithm</u>:

```
A* algorithm (U, V, C, D)

1. Let U be source and V be destination.
2. Start traversing from the U.
3. Let C = h(n) and D=g(n).
4. Let Ui=(U1,U2,.....,Un) where U1 to Un are intermediate nodes between U and V.
5. Now U => Ui for i = 1 where i = ith level nodes
    f(n) = h(n) + g(n)
6. For i > 1
if
    U => Ui is directly connected where i > 1
    f(n) = h(n) + g(n)
    where is g(n) is backtracking distance from source i.e. U.

else
    U =>  Ui is connected where i > 1
    f(n) = h(n) + g(n)
    where g(n) is distance which is summation of distance from Ui to U.

7. Total A* cost
  f(i) = ∑ from n=1 to i (f(n))
   will be summation from U to V.
```

A* search, like best-first search, evaluates a search space using a heuristic function. But A* uses both the cost of getting from the initial state to thecurrent state (*g(n)*), as well as an estimated cost (heuristic) of the path fromthe current node to the goal (*h(n)*). These are summed to the cost function *f(n)*. The A* search, unlike best-first, is both optimal and complete.The OPEN and CLOSED lists are used again to identify the frontier forsearch (OPEN list) and the nodes evaluated thus far (CLOSED). The OPENlist is implemented as a priority queue ordered in lowest *f(n)* order. Whatmakes A* interesting is that it continually re-evaluates the cost function fornodes as it re-encounters them. This allows A* to efficiently find the minimalpath from the initial state to the goal state.Let’s now look at A* at a high level and then we’ll dig further and applyit to a well-known problem. This Listingprovides the high level flow for A*.

LISTING 3.4: High-level flow for the A* search algorithm.

```
Initialize OPEN list (priority queue)
Initialize CLOSED list
Place start node on the OPEN list
Loop while the OPEN list is not empty
    Get best node (parent) from OPEN list (least f (n))
    if parent is the goal node, done
    Place parent on the CLOSED list
    Expand parent to all adjacent nodes (adj_node)

        if adj_node is on the CLOSED list
            discard adj_node and continue
        else if adj_node is on the OPEN list
            if adj_node’s g value is better than

                the OPEN.adj_node’s g value
                discard OPEN.cur_node
                calculate adj_node’s g, h and f values
                set adj_node predecessor to parent
                add adj_node to OPEN list
                continue
            end

            else
              calculate adj_node’s g, h and f values
            set adj_node predecessor to parent

            add adj_node to OPEN list
        end
    end
end loop
```

Once we find the best node from the OPEN list, we expand all of the child nodes (legal states possible from thebest node). If the new legal states are not found on either the OPEN or CLOSED lists, they are added as new nodes (setting the predecessor to thebest node, or parent). If the new node is on the CLOSED list, we discardit and continue. Finally, if the new node is on the OPEN list, but the newnode

has a better *g* value, we discard the node on the OPEN list and addthe new node to the OPEN list (otherwise, the new node is discarded, if its *g*value is worse). By re-evaluating the nodes on the OPEN list, and replacingthem when cost functions permit, we allow better paths to emerge from thestate space.
As we've defined already, A* is complete, as long as the memory supportsthe depth and branching factor of the tree. A* is also optimal, but thischaracteristic depends on the use of an *admissible* heuristic. Because A* must keep track of the nodes evaluated so far (and also the discovered nodesto be evaluated), the time and space complexity are both O(b^d).The heuristic is defined as admissible if it accurately estimates thepath cost to the goal, or underestimates it (remains optimistic). Thisrequires that the heuristic be monotonic, which means that the costnever decreases over the path, and instead monotonically increases. Thismeans that g(n) (path cost from the initial node to the current node)monotonically increases, while h(n) (path cost from the current node tothe goal node) monotonically decreases.

Program with proper indentation& comments:

Pseudo code:

```
OPEN //the set of nodes to be evaluated
CLOSED //the set of nodes already evaluated
add the start node to OPEN

loop
    current = node in OPEN with the lowest f_cost
    remove current from OPEN
    add current to CLOSED

    if current is the target node //path has been found
        return

    foreach neighbour of the current node
        if neighbour is not traversable or neighbour is in CLOSED
            skip to the next neighbour

        if new path to neighbour is shorter OR neighbour is not in OPEN
            set f_cost of neighbour
            set parent of neighbour to current
            if neighbour is not in OPEN
                add neighbour to OPEN
```

```
//Source Code

import java.util.*;

public class AStar {
    public static final int DIAGONAL_COST = 14;
    public static final int V_H_COST = 10;
```

```
static class Cell{
    int heuristicCost = 0; //Heuristic cost
    int finalCost = 0; //G+H
    int i, j;
    Cell parent;

    Cell(int i, int j){
        this.i = i;
        this.j = j;
    }

    @Override
    public String toString(){
        return "["+this.i+", "+this.j+"]";
    }
}

//Blocked cells are just null Cell values in grid
static Cell [][] grid = new Cell[5][5];

static PriorityQueue<Cell> open;

static boolean closed[][];
static int startI, startJ;
static int endI, endJ;

public static void setBlocked(int i, int j){
    grid[i][j] = null;
}

public static void setStartCell(int i, int j){
    startI = i;
    startJ = j;
}

public static void setEndCell(int i, int j){
    endI = i;
    endJ = j;
}

static void checkAndUpdateCost(Cell current, Cell t, int cost){
    if(t == null || closed[t.i][t.j])return;
    int t_final_cost = t.heuristicCost+cost;

    boolean inOpen = open.contains(t);
    if(!inOpen || t_final_cost<t.finalCost){
        t.finalCost = t_final_cost;
        t.parent = current;
        if(!inOpen)open.add(t);
    }
}

public static void AStar(){

    //add the start location to open list.
    open.add(grid[startI][startJ]);

    Cell current;

    while(true){
        current = open.poll();
        if(current==null)break;
        closed[current.i][current.j]=true;

        if(current.equals(grid[endI][endJ])){
            return;
        }

        Cell t;
        if(current.i-1>=0){
            t = grid[current.i-1][current.j];
            checkAndUpdateCost(current, t, current.finalCost+V_H_COST);

            if(current.j-1>=0){
                t = grid[current.i-1][current.j-1];
                checkAndUpdateCost(current, t, current.finalCost+DIAGONAL_COST);
            }

            if(current.j+1<grid[0].length){
                t = grid[current.i-1][current.j+1];
                checkAndUpdateCost(current, t, current.finalCost+DIAGONAL_COST);
            }
        }

        if(current.j-1>=0){
            t = grid[current.i][current.j-1];
            checkAndUpdateCost(current, t, current.finalCost+V_H_COST);
        }

        if(current.j+1<grid[0].length){
            t = grid[current.i][current.j+1];
            checkAndUpdateCost(current, t, current.finalCost+V_H_COST);
        }

        if(current.i+1<grid.length){
            t = grid[current.i+1][current.j];
            checkAndUpdateCost(current, t, current.finalCost+V_H_COST);
            if(current.j-1>=0){
                t = grid[current.i+1][current.j-1];
                checkAndUpdateCost(current, t, current.finalCost+DIAGONAL_COST);
            }

            if(current.j+1<grid[0].length){
               t = grid[current.i+1][current.j+1];
                checkAndUpdateCost(current, t, current.finalCost+DIAGONAL_COST);
            }
        }
    }
}

/*
Params :
tCase = test case No.
x, y = Board's dimensions
si, sj = start location's x and y coordinates
ei, ej = end location's x and y coordinates
int[][] blocked = array containing inaccessible cell coordinates
*/
public static void test(int tCase, int x, int y, int si, int sj, int ei, int ej, int[][] blocked){
       System.out.println("\n\nTest Case #"+tCase);
        //Reset
       grid = new Cell[x][y];
       closed = new boolean[x][y];
       open = new PriorityQueue<>((Object o1, Object o2) -> {
            Cell c1 = (Cell)o1;
            Cell c2 = (Cell)o2;

            return c1.finalCost<c2.finalCost?-1:
                    c1.finalCost>c2.finalCost?1:0;
        });
       //Set start position
       setStartCell(si, sj);  //Setting to 0,0 by default. Will be useful for the UI part

       //Set End Location
       setEndCell(ei, ej);

       for(int i=0;i<x;++i){
          for(int j=0;j<y;++j){
              grid[i][j] = new Cell(i, j);
              grid[i][j].heuristicCost = Math.abs(i-endI)+Math.abs(j-endJ);
//                   System.out.print(grid[i][j].heuristicCost+" ");
          }
//               System.out.println();
       }
       grid[si][sj].finalCost = 0;
    /*
      Set blocked cells. Simply set the cell values to null
      for blocked cells.
    */
    for(int i=0;i<blocked.length;++i){
        setBlocked(blocked[i][0], blocked[i][1]);
    }

    //Display initial map
    System.out.println("Grid: ");
     for(int i=0;i<x;++i){
         for(int j=0;j<y;++j){
            if(i==si&&j==sj)System.out.print("SO  "); //Source
            else if(i==ei && j==ej)System.out.print("DE  ");  //Destination
            else if(grid[i][j]!=null)System.out.printf("%-3d ", 0);
            else System.out.print("BL  ");
         }
         System.out.println();
     }
     System.out.println();

    AStar();
    System.out.println("\nScores for cells: ");
    for(int i=0;i<x;++i){
        for(int j=0;j<x;++j){
            if(grid[i][j]!=null)System.out.printf("%-3d ", grid[i][j].finalCost);
            else System.out.print("BL  ");
        }
        System.out.println();
    }
    System.out.println();

    if(closed[endI][endJ]){
        //Trace back the path
         System.out.println("Path: ");
         Cell current = grid[endI][endJ];
         System.out.print(current);
         while(current.parent!=null){
             System.out.print(" -> "+current.parent);
             current = current.parent;
         }
         System.out.println();
    }else System.out.println("No possible path");
}

public static void main(String[] args) throws Exception{
test(1, 5, 5, 0, 0, 3, 2, new int[][]{{0,4},{2,2},{3,1},{3,3}});
test(2, 5, 5, 0, 0, 4, 4, new int[][]{{0,4},{2,2},{3,1},{3,3}});
test(3, 7, 7, 2, 1, 5, 4, new int[][]{{4,1},{4,3},{5,3},{2,3}});
```

```
        test(1, 5, 5, 0, 0, 4, 4, new int[][]{{3,4},{3,3},{4,3}});
    }
}
```

Code complexity:

Time and space complexity are both O(b^d)

Conclusions: In this way, we have learnt about A* star search for the problems belonging to informed search space in artificial intelligence.

Assignment No 4

Aim: To implement unification algorithm

Objectives: To study & implement unification algorithm

Tools Requirement:

Hardware: - 64-bit intel i5 processor and above

Software: - 64-bit Fedora OS with in-built eclipse IDE

Theory:

The unification problem in first- order logic can be expressed as follows: Given two terms containing some variables, find, if it exists, the simplest substitution(i.e.an assignment of some term to every variable) which makes the two terms equal. The resulting substitution is called the most general unifier.

Unification, in computer science and logic, is an algorithmic process of solving equations between symbolic expressions.

Depending on which expressions (also called *terms*) are allowed to occur in an equation set (also called unification problem), and which expressions are considered equal, several frameworks of unification are distinguished. If *higher-order variables*, that is, variables representing functions, are allowed in an expression, the process is called higher-order unification, otherwise first-order unification. If a solution is required to make both sides of each equation literally equal, the process is called syntactical unification, otherwise semantical, or equational unification, or E-unification, or unification modulo theory.

A solution of a unification problem is denoted as a substitution, that is, a mapping assigning a symbolic value to each variable of the problem's expressions. A unification algorithm should compute for a given problem a complete, and minimal substitution set, that is, a set covering all its solutions, and containing no redundant members. Depending on the framework, a complete and minimal substitution set may have at most one, at most finitely many, or possibly infinitely many members, or may not exist at all. In some frameworks it is generally impossible to decide whether any solution exists. For first-order syntactical unification, Martelli and Montanari gave an algorithm that reports unsolvability or computes a complete and minimal singleton substitution set containing the so-called most general unifier.

For example, using x,y,z as variables, the singleton equation set { $cons(x,cons(x,nil)) = cons(2,y)$ } is a syntactic first-order unification problem that has the substitution { $x \mapsto 2$, $y \mapsto cons(2,nil)$ } as its only solution. The syntactic first-order unification problem { $y = cons(2,y)$ } has no solution over the set of finite terms; however, it has the single solution { $y \mapsto cons(2,cons(2,cons(2,...)))$ } over the set of infinite trees. The semantic first-order unification problem { $a{\cdot}x = x{\cdot}a$ } has each substitution of the form { $x \mapsto a{\cdot}...{\cdot}a$ } as a solution in a semigroup, i.e. if (·) is considered associative; the same problem, viewed in an abelian group, where (·) is considered also commutative, has any substitution at all as a solution. The singleton set { $a = y(x)$ } is a syntactic second-order unification problem, since y is a function variable. One solution is { $x \mapsto a$, $y \mapsto$ (identity function) }; another one is { $y \mapsto$ (constant function mapping each value to a), $x \mapsto$ *(any value)* }.

Program :

```
#include<stdio.h>
#include<conio.h>
int no_of_pred;
int no_of_arg[10];
int i,j;
char nouse;
char predicate[10];
char argument[10][10];


void unify();
void display();
void chk_arg_pred();


   void main()
   {
   char ch;
   do{
   clrscr();

            printf("\t=========PROGRAM FOR UNIFICATION=========\n");
            printf("\nEnter Number of Predicates:- [ ]\b\b");
            scanf("%d",&no_of_pred);

            for(i=0;i<no_of_pred;i++)
            {
            scanf("%c",&nouse);    //to accept "Enter" as a character
            printf("\nEnter Predicate %d:-[ ]\b\b",i+1);
            scanf("%c",&predicate[i]);
            printf("\n\tEnter No.of Arguments for Predicate %c:-[ ]\b\b",predicate[i]);
            scanf("%d",&no_of_arg[i]);
                        for(j=0;j<no_of_arg[i];j++)
                        {
                         scanf("%c",&nouse);
                         printf("\n\tEnter argument %d:( )\b\b",j+1);
                         scanf("%c",&argument[i][j]);
                        }
            }

            display();
            chk_arg_pred();
            getch();
            flushall();
            printf("Do you want to continue(y/n): ");
            scanf("%c",&ch);
```

```
    }while(ch=='y');
}

void display()
{
    printf("\n\t=======PREDICATES ARE======");
         for(i=0;i<no_of_pred;i++)
         {
          printf("\n\t%c(",predicate[i]);
                     for(j=0;j<no_of_arg[i];j++)
                     {
                     printf("%c",argument[i][j]);
                     if(j!=no_of_arg[i]-1)
                                 printf(",");
                     }
          printf(")");
         }
}

void chk_arg_pred()
{
int pred_flag=0;
int arg_flag=0;

/*======Checking Prediactes========*/
         for(i=0;i<no_of_pred-1;i++)
         {
                     if(predicate[i]!=predicate[i+1])
                     {
                     printf("\nPredicates not same..");
                     printf("\nUnification cannot progress!");
                     pred_flag=1;
                     break;
                     }
         }
/*=====Chking No of Arguments====*/
  if(pred_flag!=1)
  {
         for(i=0;i<no_of_arg[i]-1;i++)
         {
                     if(no_of_arg[i]!=no_of_arg[i+1])
                     {
                     printf("\nArguments Not Same..!");
                     arg_flag=1;
                     break;
                     }
         }
  }
         if(arg_flag==0&&pred_flag!=1)
```

```
                    unify();

   }
/*==========UNIFY FUNCTION=========*/
   void unify()
   {
          int flag=0;
          for(i=0;i<no_of_pred-1;i++)
          {
              for(j=0;j<no_of_arg[i];j++)
              {
                    if(argument[i][j]!=argument[i+1][j])
                    {
                      if(flag==0)
                      printf("\n\t======SUBSTITUTION IS======");
                    printf("\n\t%c/%c",argument[i+1][j],argument[i][j]);
                     flag++;
                    }
             }
          }
          if(flag==0)
          {         printf("\nArguments are Identical...");
                    printf("\nNo need of Substitution\n");
          }
   }
```

Output :

```
Enter Number of Predicates:- [2]
Enter Predicate 1:-[P]
        Enter No.of Arguments for Predicate P:-[2]
        Enter argument 1:(a)
        Enter argument 2:(b)
Enter Predicate 2:-[P]
        Enter No.of Arguments for Predicate P:-[2]
        Enter argument 1:(c)
        Enter argument 2:(b)
        =======PREDICATES ARE======
        P(a,b)
        P(c,b)
        ======SUBSTITUTION IS======
        c/a
Do you want to continue(y/n): y
        =========PROGRAM FOR UNIFICATION=========
Enter Number of Predicates:- [2]
Enter Predicate 1:-[P]
        Enter No.of Arguments for Predicate P:-[3]
        Enter argument 1:(a)
        Enter argument 2:(b)
        Enter argument 3:(c)
Enter Predicate 2:-[P]
        Enter No.of Arguments for Predicate P:-[2]
        Enter argument 1:(d)
        Enter argument 2:(r)
        =======PREDICATES ARE======
        P(a,b,c)
        P(d,r)
Arguments Not Same..!Do you want to continue(y/n): y
        =========PROGRAM FOR UNIFICATION=========
Enter Number of Predicates:- [2]
Enter Predicate 1:-[P]
        Enter No.of Arguments for Predicate P:-[2]
        Enter argument 1:(1)
        Enter argument 2:(3)
       Enter Predicate 2:-[Q]
        Enter No.of Arguments for Predicate Q:-[2]
        Enter argument 1:(3)
        Enter argument 2:(4)
        =======PREDICATES ARE======
        P(1,3)
        Q(3,4)
Predicates not same..
Unification cannot progress!
```

Case study (Any example) :

apply the unification algorithm to the following two clauses and show how it progresss towards finding a most general unifier or concluding that the clause is not unifiable?
{P(x,y),P(y,f(z))},{P(g(x),y),P(y,y),P(u,f(w))}
for the clause {P(x,y),P(y,f(z))} the substitution as sub=sub[x/y][y/f(z)] and the most general unifier as Lsub={P(f(z),f(z))}
for the second clause {P(g(x),y),P(y,y),P(u,f(w))} follow the unifier algorithm step1:sub=sub[y/g(x)] and has a result as {P(g(x),g(x)}
 step2:sub=sub[u/g(x)] and has a result as {P(g(x),f(w)}
step3:since non of "f" and "g" is variable. so output non-unifiable

Mathematical modeling:

Consider expressions built up from variables and constants using function symbols.
 If v_1, ..., $v_n$are variables and t_1, ..., $t_n$are expressions, then a set of mappings between variables and expressions $\{t_1 \mid v_1, \ldots, t_n \mid v_n\}$is called a substitution.
If $\eta = \{t_1 \mid v_1, \ldots, t_n \mid v_n\}$and $E$is an expression, then $E\eta$is called an instance of Eif it is received from E by simultaneously replacing all occurrences of v_i(for $0 \le i \le n$) by the respective t_i.
If $\eta = \{t_1 \mid v_1, \ldots, t_n \mid v_n\}$and $\theta = \{u_1 \mid x_1, \ldots, u_n \mid x_m\}$are two substitutions, then the composition of $\eta$and θ (denoted $\eta * \theta$) is obtained from $\{t_1\theta \mid v_1, \ldots, t_n\theta \mid v_n, u_1 \mid x_1, \ldots, u_n \mid x_m\}$by removing all elements $t_i\theta \mid v_i$

such that $t_i\ \theta = v_i$and all elements $u_i \mid x_i$ such that $x_i$is one of v_1, ..., v_n.
A substitution $\eta$is called a unifier for the set of expressions $\{E_1, \ldots, E_n\}$if $E_1\ \eta = E_2\ \eta = \ldots = E_n\ \eta$. A unifier $\eta$for the set of expressions $\{E_1, \ldots, E_n\}$is called the most general unifier if, for any other unifier for the same set of expressions θ, there is yet another unifier ιsuch that $\theta = \eta * \iota$.
A unification algorithm takes a set of expressions as its input. If this set is not unifiable, the algorithm terminates and yields a negative result. If there exists a unifier for the input set of expressions, the algorithm yields the most general unifier for this set of expressions. The unification algorithm serves as a tool for the resolution principle.

Optimized Code:

```
function Unify(f1, f2) returns fs or failure
f1.real ← real contents of f1, *(f1.ptr) if ptr != null
f2.real ← real contents of f2, *(f2.ptr) if ptr != null
if (f1.real is null) { f1.ptr ← f2; return f2 }
else if (f2.real is null) { f2.ptr ← f1; return f1 }
else if (f1.real == f2.real) {f1.ptr ← f2; return f2 }
else if (f1.real is complex and f2.real is complex) {
  f2.ptr ← f1
  foreach ftr2 in f2.real {
    ftr1 ← find/create feature ftr2 in f1.real
    if Unify(ftr2.value,ftr1.value) fails {return failure}
  return f1
else return failure
```

Code complexity: O(n^2)

Conclusions: In this way we have studied and implemented unification algorithm.

Assignment No 5

Title: Study and implementation of Navie Bays Classification algorithm.
Aim: Implement Naive Bayes to predict the work type for a person with following parameters: age: 30, Qualification: MTech, Experience: 8

Following table provides the details of the available data:

Work Type	Age	Qualification	Experience
Consultancy	30	Ph.D.	9
Service	21	MTech.	1
Research	26	MTech.	2
Service	28	BTech.	10
Consultancy	40	MTech.	14
Research	35	Ph.D.	10
Research	27	BTech.	6
Service	32	MTech.	9
Consultancy	45	Btech.	17
Research	36	Ph.D.	7

Objectives:

1. To study the concept of Machinelearning .
2. To study the Principle of Navie Bays probabilistic classification Algorithm.
3. To study the different applications of The algorithm
4. To study the pros and cons of algorithm.
5. To predict the work type for above mentioned problem by usinnavie bays algorithm

Tools Requirement: Not Required
Hardware: - any PC with minimum of 512 RAM
Software: - C, C++, Java.

Theory:
The Bayesian Classification represents a supervised learning method as well as a statistical method for classification.
Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data.

About Naive Bayes algorithm

- Statistical method for classification.
- Supervised Learning Method.
- Assumes an underlying probabilistic model, the Bayes theorem.
- Can solve problems involving both categorical and continuous valued attributes.
- Named after *Thomas Bayes, who proposed the Bayes Theorem.*

Issues of navie Bays

Violation of Independence Assumption

- Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered "naive."

Zero conditional probability Problem

- ✓ If a given class and feature value never occur together in the training set then the frequency-based probability estimate will be zero.
- ✓ This is problematic since it will wipe out all information in the other probabilities when they are multiplied.
- ✓ It is therefore often desirable to incorporate a small-sample correction in all probability estimates such that no probability is ever set to be exactly zero.

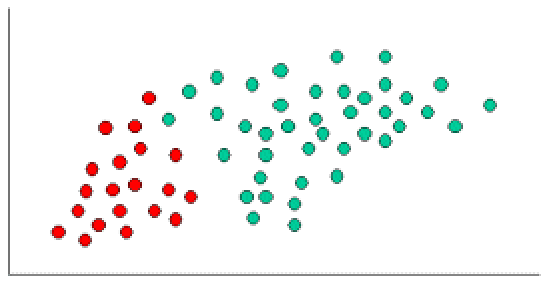
Uses of Naive Bayes classification:

- Naive Bayes text classification
- Spam filtering
- Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering
- Emotional Modeling
- Web mining

The Bayes Theorem

- The Bayes Theorem:
    - P(H|X)= P(X|H) P(H)/ P(X)
    - P(H|X) : (Posterior Probability of H
- P(H) : Prior Probability of H
- P(X|H) : (Posterior Probability of X
- P(X) : Prior Probability of X

Demonstration of the concept of Naïve Bayes Classification, consider the example given below:



As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

Prior Probability of GREEN: number of GREEN objects / total number of objects

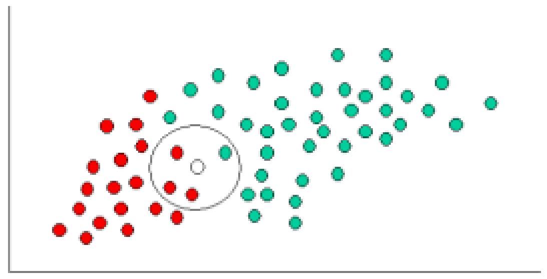
Prior Probability of RED: number of RED objects / total number of objects

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:

Prior Probability for GREEN: 40 / 60

Prior Probability for RED: 20 / 60

Having formulated our prior probability, we are now ready to classify a new object (WHITE circle in the diagram below). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:



$$\text{Likelihood of } X \text{ given GREEN} \propto \frac{\text{Number of GREEN in the vicinity of } X}{\text{Total number of GREEN cases}}$$

$$\text{Likelihood of } X \text{ given RED} \propto \frac{\text{Number of RED in the vicinity of } X}{\text{Total number of RED cases}}$$

From the illustration above, it is clear that Likelihood of X given GREEN is smaller than Likelihood of X given RED, since the circle encompasses 1GREEN object and 3RED ones. Thus:

$$\text{Probability of } X \text{ given GREEN} \propto \frac{1}{40}$$

$$\text{Probability of } X \text{ given RED} \propto \frac{3}{20}$$

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).

$$\textit{Posterior probability of } X \textit{ being GREEN} \propto$$
$$\textit{Prior probability of GREEN} \times \textit{Likelihood of } X \textit{ given GREEN}$$
$$= \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$
$$\textit{Posterior probability of } X \textit{ being RED} \propto$$
$$\textit{Prior probability of RED} \times \textit{Likelihood of } X \textit{ given RED}$$
$$= \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Finally, we classify X as RED since its class membership achieves the largest posterior probability.

posterior probability=conditional probability·prior probabilityevidence

Prior Probablities : In the context of pattern classification, the prior probabilities are also called classpriors, which describe ”the general probability of encountering a particular class.” I

Program:

Steps followed to program the given problem.

1. **Prior Probabilities of consultancy, Service and research to be calculated.**
   P(consultancy)= 3/10
   P(service)=3/10
   P(research)=4/10

2. **Mean and variances for age and experience is calculated for service ,research and consultancy by using mean and variance formula .**

   Mean(age for consultancy)
   Std. deviation (age for consultancy)

   Mean(age for service)
   Std. deviation (age for service)

   Mean(age for research)
   Std. deviation (age for research)

3. By using Mean and variance probabilistic distribution is calculated

   P(age/consultancy) (probability distribution of age given consultancy)=

   1/ (Sqrtof(2 *pi*std.deviation age))exp (-(30(given age)))- square of (mean age for consultancy))/(2 * std. Deviation)

   P(exp/consultancy)(probability distribution of exp given consultancy) =1/Sqrtof(2 *pi*std.deviation of exp)exp (-(30(given exp)- square of (meanexp for consultancy))/(2 * std. Deviation)

   P(Mtech/consultancy) (probability of mtech given consultancy) = 1/3

   ****
   P(age/service)(probability distribution of age given service) **=**
   1/Sqrtof(2 *pi*std.deviation)exp (-(30(given age)- square of (mean age for service))/(2 * std. Deviation)

   P(exp/service) (probability distribution of exp given service)=1/Sqrt of(2 *pi*std.deviation)exp (-(30(given age)- square of (meanexp for service))/(2 * std. Deviation)
   P(Mtech/service)(probability of mtech given service)=2/3

   ****
   P(age/research)(probability distribution of age given research)=
   1/Sqrtof(2 *pi*std.deviation)exp (-(30(given age)- square of (mean age for research))/(2 * std. Deviation)

   P(exp/research)(probability distribution of exp given research) =
   1/Sqrtof(2 *pi*std.deviation)exp (-(30(given age)- square of (meanexp for research))/(2 * std. Deviation)

   P(Mtech/research)(probability of mtech given research)=1/4
   ****
4. **Calculate the probability density for consultancy , service and research by using following formula as experience and age is continuous variables.**

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

μ_{ji} : mean (avearage)of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

5. Calculate Evidence as

Evidence =P(consultancy) *P(age/consultancy) *P(exp/consultancy) *P((mtech/consultancy)
+
P(Service) *P(age/service) *P(exp/service) *P((mtech/service)
+
P(research) *P(age/research) *P(exp/research) *P((mtech/research)

6. CalculatePosterior Probability of consultancy , Service and Research by using Following formula

Posterior Probability P(Consultancy) =
P(consultanc y) *P(age/consultancy)*P(exp/consultancy)* P(Mtech/Consultancy)/Evidence

Posterior Probability P(service) =
P(service) *P(age/service)*P(exp/service)* P(Mtech/service)/Evidence

Posterior Probability P(Research) =
P(Research) *P(age/Research)*P(exp/Research)* P(Mtech/Research)/Evidence

For classification the largest probability distribution is considered.

C source code for Navie Bays Classification algorithm

```
#include<stdio.h>
#include<string.h>
#include<math.h>
/*******************************************************/
struct record
{
        char WT[10];
        float Age;
        char Qua[10];
        float Exp;
};
/*******************************************************/
struct record rec[10];
struct record con[4];
struct record service[4];
struct record research[4];
/*******************************************************/
int cc=0,sc=0,rc=0,sample_age=0,sample_exp=0;
float m_age[3],m_exp[3];
float v_age[3],v_exp[3];
float p_a_c=0,p_a_s=0,p_a_r=0;
float p_e_c=0,p_e_s=0,p_e_r=0;
/*******************************************************/
void accept(struct record rec[],int n)
{
        int i;
        for(i=0;i<n;i++)
        {
                printf("\nEnter the work type : ");
                scanf("%s",&rec[i].WT);

                printf("\nEnter the Age : ");
                scanf("%f",&rec[i].Age);

                printf("\nEnter the Qualification : ");
                scanf("%s",&rec[i].Qua);

                printf("\nEnter the Experience : ");
                scanf("%f",&rec[i].Exp);
        }

}
/****************************************************************/
void sort(struct record rec[],int n)
{
        int i,j=0,k=0,l=0,lcc=0,lsc=0,lrc=0,x;

        for(i=0;i<n;i++)
        {
                x=strcmp(rec[i].WT,"consultancy");
                if(x==0)
                {
                        strcpy(con[j].WT,rec[i].WT);

                        con[j].Age=rec[i].Age;

                        strcpy(con[j].Qua,rec[i].Qua);

                        con[j].Exp=rec[i].Exp;

                        lcc++;
                        j++;
                        continue;
                }
                else
                {
                        continue;
                }
        }

        for(i=0;i<n;i++)
        {
                x=strcmp(rec[i].WT,"research");
                if(x==0)
                {
                        strcpy(research[l].WT,rec[i].WT);

                        research[l].Age=rec[i].Age;
                        strcpy(research[l].Qua,rec[i].Qua);

                        research[l].Exp=rec[i].Exp;
                        lrc++;
                        l=l+1;
                        continue;
                }
                else
                {
                        continue;
                }
        }

        for(i=0;i<n;i++)
        {
                x=strcmp(rec[i].WT,"service");
                if(x==0)
                {
                        strcpy(service[k].WT,rec[i].WT);

                        service[k].Age=rec[i].Age;

                        strcpy(service[k].Qua,rec[i].Qua);

                        service[k].Exp=rec[i].Exp;

                        lsc++;
                        k=k+1;
                        continue;
                }
                else
                {
                        continue;
                }
        }

        cc=lcc;
        sc=lsc;
        rc=lrc;
}
/***********************************************************/
void mean(int n)
{
        int i;
        float t_age=0,t_exp=0;

        int lcc=0,lsc=0,lrc=0;
        lcc=cc;
        lsc=sc;
        lrc=rc;

        for(i=0;i<lcc;i++)
        {
                t_age=t_age+con[i].Age;
        }
        m_age[0]=(t_age/lcc);

        t_age=0;
        for(i=0;i<lsc;i++)
        {
                t_age=t_age+service[i].Age;
        }
        m_age[1]=(t_age/lsc);

        t_age=0;
        for(i=0;i<lrc;i++)
        {
                t_age=t_age+research[i].Age;
        }
        m_age[2]=(t_age/lrc);
/*********************************************************/
        t_exp=0;
        for(i=0;i<lcc;i++)
        {
                t_exp=t_exp+con[i].Exp;
        }
        m_exp[0]=(t_exp/lcc);

        t_exp=0;
        for(i=0;i<lsc;i++)
        {
                t_exp=t_exp+service[i].Exp;
        }
        m_exp[1]=(t_exp/lsc);

        t_exp=0;
        for(i=0;i<lrc;i++)
        {
                t_exp=t_exp+research[i].Exp;
        }
        m_exp[2]=(t_exp/lrc);
}
/*********************************************************/
void variance(int n)
{
        int i;
        float t_age=0,t_exp=0;
        int lcc=0,lsc=0,lrc=0;
        float t_v=0;

        lcc=cc;
        lsc=sc;
        lrc=rc;

        t_v=0;
        for(i=0;i<lcc;i++)
        {
                t_age=0;
                t_age=m_age[0]-con[i].Age;
                t_age=t_age*t_age;
                t_v=t_v+t_age;
        }
        v_age[0]=(t_v/(lcc-1));

        t_v=0;
        for(i=0;i<lsc;i++)
        {
                t_age=0;
                t_age=m_age[1]-service[i].Age;
                t_age=t_age*t_age;
                t_v=t_v+t_age;
        }
        v_age[1]=(t_v/(lsc-1));

        t_v=0;
        for(i=0;i<lrc;i++)
        {
                t_age=0;
                t_age=m_age[2]-research[i].Age;
                t_age=t_age*t_age;
                t_v=t_v+t_age;
        }
        v_age[2]=(t_v/(lrc-1));
/**********************************************************/
        t_v=0;
        for(i=0;i<lcc;i++)
        {
                t_exp=0;
                t_exp=m_exp[0]-con[i].Exp;
                t_exp=t_exp*t_exp;
                t_v=t_v+t_exp;
        }
        v_exp[0]=(t_v/(lcc-1));

        t_v=0;
        for(i=0;i<lsc;i++)
```

```
	{
		t_exp=0;
		t_exp=m_exp[1]-service[i].Exp;
		t_exp=t_exp*t_exp;
		t_v=t_v+t_exp;
	}
	v_exp[1]=(t_v/(lsc-1));

	t_v=0;
	for(i=0;i<lrc;i++)
	{
		t_exp=0;
		t_exp=m_exp[2]-research[i].Exp;
		t_exp=t_exp*t_exp;
		t_v=t_v+t_exp;
	}
	v_exp[2]=(t_v/(lrc-1));

}
/**********************************************************/
void prob()
{
	int i;
	float pi=6.2832;
	float constant=0,e_val=0,e_val1=0,e_val2=0;
	float prob_age[3],prob_exp[3];

	pi=sqrt(pi);
	constant=(1/pi);

	for(i=0;i<3;i++)
	{
		e_val=e_val1=e_val2=0;
		e_val=(-((sample_age-m_age[i])*(sample_age-m_age[i]))/((2*v_age[i])));
		e_val1=(1/sqrt(v_age[i]));
		e_val2=exp(e_val);
		prob_age[i]=constant*e_val1*e_val2;
	}
	p_a_c=prob_age[0];
	p_a_s=prob_age[1];
	p_a_r=prob_age[2];
/**********************************************************/
	for(i=0;i<3;i++)
	{
		e_val=e_val1=e_val2=0;
		e_val=(-((sample_exp-m_exp[i])*(sample_exp-m_exp[i]))/((2*v_exp[i])));
		e_val1=(1/sqrt(v_exp[i]));
		e_val2=exp(e_val);
		prob_exp[i]=constant*e_val1*e_val2;
```

```
        }
        p_e_c=prob_exp[0];
        p_e_s=prob_exp[1];
        p_e_r=prob_exp[2];
}
/**********************************************************/
void main()
{
        int n,i,x;
        char sample_qua[10];
        float p_c,p_s,p_r,counter=0;
        float p_q_c=0,p_q_s=0,p_q_r=0;
        float evidence;
        float lcc,lsc,lrc,fn;
        float pos_c=0,pos_s=0,pos_r=0;

        printf("How many records you want to add to list : ");
        scanf("%d",&n);
        accept(rec,n);
        printf("\nEnter the sample age : ");
        scanf("%d",&sample_age);
        printf("\nEnter the sample experience : ");
        scanf("%d",&sample_exp);
        printf("\nEnter the sample qualifiction : ");
        scanf("%s",&sample_qua);

        sort(rec,n);
        mean(n);
        variance(n);
        prob();

        lcc=cc;
        lsc=sc;
        lrc=rc;
        fn=n;

        p_c=(lcc/fn);
        p_s=(lsc/fn);
        p_r=(lrc/fn);

        for(i=0;i<cc;i++)
        {
                x=strcmp(con[i].Qua,sample_qua);
                if(x==0)
                if((strcmp(con[i].Qua,sample_qua))==0)
                {
                        counter++;
                        continue;
                }
```

```
		else
			continue;
	}

	p_q_c=(counter/cc);
	counter=0;
	for(i=0;i<sc;i++)
	{
		x=strcmp(service[i].Qua,sample_qua);
		if(x==0)
		if((strcmp(service[i].Qua,sample_qua))==0)
		{
			counter++;
			continue;
		}
		else
			continue;
	}

	p_q_s=(counter/sc);
	counter=0;
	for(i=0;i<rc;i++)
	{
		x=strcmp(research[i].Qua,sample_qua);
		if(x==0)
		{
			counter++;
			continue;
		}
		else
			continue;
	}
	p_q_r=(counter/rc);

	evidence=((p_c*p_a_c*p_q_c*p_e_c)+(p_s*p_a_s*p_q_s*p_e_s)+(p_r*p_a_r*p_q_r*p_e_r));
	printf("\n\nEvidence : %f",evidence);
/*********************************************************/
	pos_c=((p_c*p_a_c*p_q_c*p_e_c)/evidence);
	pos_s=((p_s*p_a_s*p_q_s*p_e_s)/evidence);
	pos_r=((p_r*p_a_r*p_q_r*p_e_r)/evidence);
	printf("\n\nPosterior Probability of Consultancy : %f",pos_c);
	printf("\n\nPosterior Probability of Service : %f",pos_s);
	printf("\n\nPosterior Probability of Research : %f",pos_r);

	if((pos_c>pos_s)&&(pos_c>pos_r))
	{
printf("\n\nWork type of person will be consultancy and the probability will be %f .\n\n",pos_c);
	}
	elseif(pos_s>pos_r)
```

```
            {
printf("\n\nWork type of person will be service and the probability will be %f .\n\n",pos_s);
            }
            else
printf("\n\nWork type of person will be research and the probability will be %f .\n\n",pos_r);
}
/*******************************************************/
```

Mathematical modeling:
Naviebayes' theorem is stated mathematically as the following equation:

$$P(A\ B) = \frac{P(B|A)P(A)}{P(B)}$$

where***A***and***B***are events.

- ***P*(*A*)** and ***P*(*B*)** are the probabilities of***A***and***B***independent of each other.
- ***P*(*A*|*B*)**, a*conditional probability*, is the probability of***A***given that***B***is true.
- ***P*(*B*|*A*)**, is the probability of ***B*** given that ***A*** is true.

- D : Set of tuples
  - Each Tuple is an 'n' dimensional attribute vector
  - X : $(x_1,x_2,x_3,\dots\ x_n)$
  - where x_i is the value of attribute A_i
- Let there are 'm' Classes : $C_1,C_2,C_3\dots C_m$
- Bayesian classifier predicts X belongs to Class C_i iff
  - $P(C_i|X) > P(C_j|X)$ for $1 <= j <= m$, $j \neq i$
- Maximum Posteriori Hypothesis
  - $P(C_i|X) = \dfrac{P(X|C_i)\ P(C_i)}{P(X)}$
  - Maximize $P(X|C_i)\ P(C_i)$ as $P(X)$ is constant

- With many attributes, it is computationally expensive to evaluate $P(X|C_i)$
- Naïve Assumption of "class conditional independence"

$$P(X \mid C_i) = P(x_1, x_2, \dots, x_n \mid C_i)$$

$$= P(x_1 \mid C_i) * P(x_2 \mid C_i) * \dots\dots\dots * P(x_n \mid C_i)$$

$$= \prod_{k=1}^{n} P(x_k \mid C_i)$$

To compute, $P(x_k|C_i)$

- A_k is categorical:

$$P(x_k|C_i) = \frac{\textit{the number of tuples of class } C_i \textit{ in } D \textit{ having the value } x_k \textit{ for } A_k}{\textit{the number of tuples of class } C_i \textit{ in } D.}$$

- A_k is continuous:

  A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

Advantages :

- ➢ Easy to implement
- ➢ Requires a small amount of training data to estimate the parameters
- ➢ Good results obtained in most of the cases

Disadvantages:

- ➢ Assumption: class conditional independence, therefore loss of accuracy
- ➢ Practically, dependencies exist among variables

E.g., hospitals: patients: Profile: age, family history, etc.
Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.

- ➢ Dependencies among these cannot be modelled by Naïve Bayesian Classifier

Testing :

Code is tasted for its robustness.

Positive testing :The given table is used to predict the type of service .

Input ;> age-30, qualification -mtech,exp-8 is provided to predict work type

Output :->service

For negative testing : The irrelevant input is provided .

It will take the (values out of bound) irrelevant values and try to predict based on it. But the classification will not be appropriate .

Invalid input is also given (such as negative values, out of range values)

Error message will be shown.

Conclusions:

- ✓ The naive Bayes model is tremendously appealing because of its simplicity, elegance, and robustness.
- ✓ It is one of the oldest formal classification algorithms, and yet even in its simplest form it is often surprisingly effective.
- ✓ It is widely used in areas such as text classification and spam filtering.

References :

- http://en.wikipedia.org/wiki/Naive_Bayes_classifier
- http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/mlbook/ch6.pdf
- Data Mining: Concepts and Techniques, 3rd Edition, Han & Kamber & Pei ISBN: 9780123814791

Assignment No 6

Aim: Design an application for Pedometer related to Android OS.

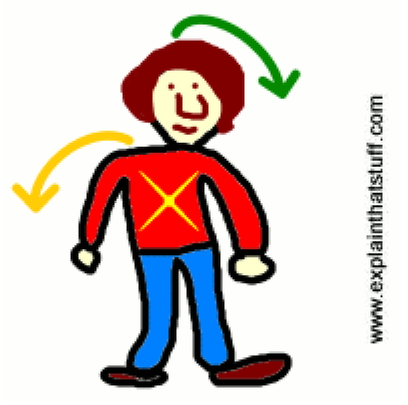
Tools:
Eclipse, JDK, Android SDK, IntelliJIDEA.

Theory:

How far do you walk in a typical day? You might be surprised at the mileage you clock up pottering round the house, nipping out to the shops, and strolling to work or school. Walking a little bit more is a great way to improve your health and fitness; for some people, this kind of exercise is prescribed by their physician as a way to lose weight or get well again after an operation. How can you keep track of your walking without measuring from a map? Simple! Just wear a handy little gadget called a pedometer that counts each step you make. Ever wondered how they work? Let's take a closer look!

Photo: A step in the right direction? Wearing a simple pedometer like this on your waist can encourage you to take more exercise. This one also has an alarm clock, stop watch, and calorie converter.

How does a pedometer work?



Suppose I give you the job of building a little gadget that will measure how far you walk in a day. Sounds like a tricky task to me. You could use something like a click wheel (a large wheel you roll over the ground that clicks each time it turns one complete circuit), but rough or muddy ground is going to cause problems and it's going to have a job measuring stairs.

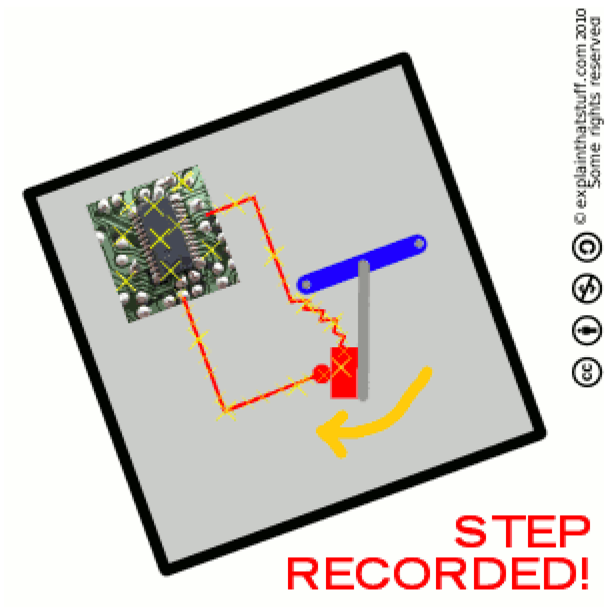
Okay, so let's redefine the problem by considering what walking involves. Every time you walk, your body tilts to one side and you swing a leg forward. Then your body tilts the other way and you swing the other leg forward too. Each tilt of the hips and shift of the legs is a step. Assuming each step is pretty much the same length, all we need to do is count the number of steps we make in a day, by counting the number of times our body tilts from side to side. We can then multiply the number of steps by the length of each one to figure out the overall distance walked. This is pretty much how a pedometer works.

Photo: Pedometers can measure your steps because your body swings from side to side as you walk. Each swing counts as one step. Multiplying the number of "swings" by the average length of your steps tells you how far you've gone.

Mechanical pedometers

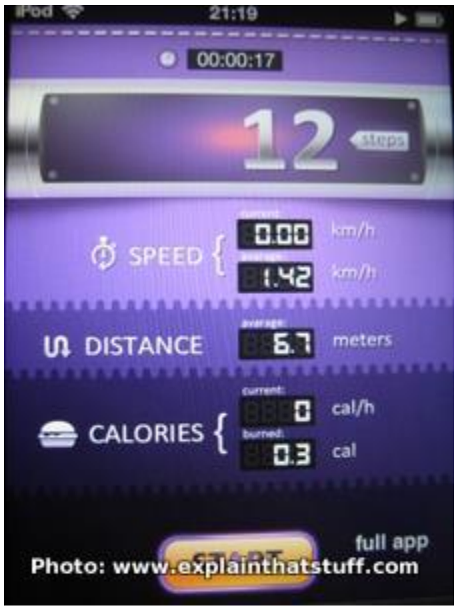
Early pedometers were entirely mechanical and they worked a bit like pendulum clocks (the ones with a swinging bar powered by a slowly falling weight). As the pendulum rocks back and forth, a kind of see-saw lever called an escapement flicks up and down and a gear wheel inside the clock (which counts seconds) advances by one position. So a pendulum clock is really a mechanism that counts seconds. The original pedometers used a swinging pendulum to count steps and displayed the count with a pointer moving round a dial (a bit like an analog watch). You fixed them on your waist and, every time you took a step, the pendulum swung to one side then back again, causing a gear to advance one position and moving the hand around the dial.

Electronic pedometers



Modern pedometers work in a very similar way but are partly electronic. Open one up and you'll find a metal pendulum (a hammer with a weight on one end) wired into an electronic counting circuit by a thin spring. Normally the circuit is open and no electric current flows through it. As you take a step, the hammer swings across and touches a metal contact in the center, completing the circuit and allowing current to flow. The flow of current energizes the circuit and adds one to your step count. As you complete the step, the hammer swings back again (helped by the spring) and the circuit is broken, effectively resetting the pedometer ready for the next step. The pedometer shows a count of your steps on an LCD display; most will convert the step count to an approximate distance in miles or kilometers (or the number of calories you've burned off) at the push of a button. Note that in some pedometers, the hammer-pendulum circuit works the opposite way: it's normally closed and each step makes it open temporarily.

Artwork: In this common design of pedometer, there's an electric circuit inside (red path) that is alternately broken and completed as you make steps. When the pedometer tilts to the left, the circuit is completed and a step is counted by an electronic circuit (top left). When it tilts the other way, the circuit is broken and reset ready to count the next step. Note how the hammer becomes part of the circuit: it has an electrical contact at one end wired to the circuit by a light metal spring (zig-zag red line). Other pedometers work in different ways, but most of the cheaper ones use a moving hammer and interruptible circuit in broadly the same way.



More sophisticated pedometers (including some of the really good ones made by Omron) work entirely electronically and, since they have no moving parts, tend to be longer-lasting, more reliable, and considerably more accurate. They dispense with the swinging pendulum-hammer and measure your steps with two or three accelerometers instead. These are microchips arranged at right angles that detect minute changes in force as you move your legs. Since accelerometers are often built into gadgets like cellphones, it's increasingly common to find these sorts of things offering to count your steps for you too (there are plenty of pedometer apps for the iPhone, for example). GPS satellite navigation devices can also figure out how far you've walked or run, but they do it by calculating from satellite signals rather than counting steps.

Photo: There are lots of pedometer apps for cellphones that make use of built-in accelerometers. This one's called Pedometer 24/7, runs on the Apple iPhone and iPod Touch, and seems to get good ratings from its users. As you can see here, it displays steps, average speed, total distance, and calories burned. You have to key in your height and weight to start, to give it a rough idea how long your steps are, and you can fine tune the sensitivity as well.

How accurate is a pedometer?

Counting steps with a pedometer sounds super-scientific, but you need to remember that it's only an approximate measurement. Not all your steps will be correctly counted and some false movements (jolts in the road as you ride in a car, for example) might be counted as steps too. Don't take the count too seriously; assume that it's in error by least 10 percent (the best electronic pedometers claim 5 percent accuracy).

For a pedometer to work correctly, you need to fix it to your waist—and not put it in your pocket—because a pedometer needs to detect the side-to-side tilting motion of your body to register each step correctly. Most devices come with a belt clip, making it reasonably easy to attach them properly. Some pedometers have a screw you can turn to alter the tension of the swinging pendulum-hammer inside them so it will register your steps correctly. If you're running, you might need to adjust it slightly differently compared to walking, for example, because your steps will likely be a different length.

Program:

```
package com.appsrox.remindme;

import java.util.Calendar;
import java.util.Date;

import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ListActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.SimpleCursorAdapter;
import android.widget.SimpleCursorAdapter.ViewBinder;
import android.widget.TextView;
import android.widget.Toast;

import com.appsrox.remindme.model.Alarm;
import com.appsrox.remindme.model.AlarmMsg;
public class MainActivity extends ListActivity {

        private SQLiteDatabase database;
        private AlertDialog alertDialog;

        public final Calendar cal = Calendar.getInstance();
        public final Date mDate = new Date();
        private String[] monthArr;
        private AlarmMsg alarmMsg = new AlarmMsg();

  @Override
  public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        database = RemindMe.database;
        monthArr = getResources().getStringArray(R.array.spinner3_arr);

        //register context menu
                registerForContextMenu(getListView());

    }



        @SuppressWarnings("deprecation")
        private Cursor createCursor() {
                Cursor c = RemindMe.dbHelper.getListNotifications(database);
                startManagingCursor(c);
                return c;
        }

    @Override
        protected void onResume() {
                super.onResume();

                @SuppressWarnings("deprecation")
                SimpleCursorAdapter adapter = new SimpleCursorAdapter(
                                this,
                                R.layout.row,
                                createCursor(),
                                new String[]{Alarm.COL_NAME, AlarmMsg.COL_DATETIME,
AlarmMsg.COL_DATETIME, AlarmMsg.COL_DATETIME, AlarmMsg.COL_DATETIME},
                                new int[]{R.id.msg_tv, R.id.year_tv, R.id.month_tv, R.id.date_tv,
R.id.time_tv});

                adapter.setViewBinder(new ViewBinder() {
                        @SuppressWarnings("deprecation")
                        @Override
                        public boolean setViewValue(View view, Cursor cursor, int columnIndex) {
                                if (view.getId() == R.id.msg_tv)
                                {
                                        return false;
                                }
                                TextView tv = (TextView)view;
                                long time = cursor.getLong(columnIndex);
                                mDate.setTime(time);
                                switch(view.getId()) {
                                case R.id.year_tv:
                                        tv.setText(String.valueOf(mDate.getYear() + 1900));
                                        break;
                                case R.id.month_tv:
                                        tv.setText(monthArr[mDate.getMonth()+1]);
```

```
					break;
				case R.id.date_tv:
					tv.setText(String.valueOf(mDate.getDate()));
					break;
				case R.id.time_tv:
					long now = System.currentTimeMillis();
					String txt = RemindMe.showRemainingTime() ?
Util.getRemainingTime(time, now) : Util.getActualTime(mDate.getHours(), mDate.getMinutes());
					if (TextUtils.isEmpty(txt)) txt =
Util.getActualTime(mDate.getHours(), mDate.getMinutes());
					tv.setText(txt);

					RelativeLayout parent = (RelativeLayout)tv.getParent();
					TextView tv2 = (TextView) parent.findViewById(R.id.msg_tv);
					if (time < now) tv2.setTextColor(Color.parseColor("#555555"));
					else tv2.setTextColor(Color.parseColor("#587498"));
					break;
				}
				return true;
			}
		});
		setListAdapter(adapter);

	}
	public void onClick(View v) {
	startActivity(new Intent(this, AddAlarmActivity.class));
	}

	@Override
	public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
		if (v.getId() == android.R.id.list) {
			getMenuInflater().inflate(R.menu.context_menu, menu);
			menu.setHeaderTitle("Choose an Option");
			menu.setHeaderIcon(R.drawable.ic_dialog_menu_generic);

			AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) menuInfo;
			alarmMsg.setId(info.id);
			alarmMsg.load(database);
			if (alarmMsg.getDateTime() < System.currentTimeMillis())
				menu.removeItem(R.id.menu_edit);
		}
	}

	@Override
	public boolean onContextItemSelected(MenuItem item) {
		AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
item.getMenuInfo();
```

```
                RemindMe.dbHelper.cancelNotification(database, info.id, false);

                Intent cancelThis = new Intent(this, AlarmService.class);
                cancelThis.putExtra(AlarmMsg.COL_ID, String.valueOf(info.id));
                cancelThis.setAction(AlarmService.CANCEL);
                startService(cancelThis);

                //Refresh adapter
                SimpleCursorAdapter adapter = (SimpleCursorAdapter) getListAdapter();
            adapter.getCursor().requery();
            adapter.notifyDataSetChanged();

            return true;
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
            openContextMenu(v);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.options_menu, menu);
   return true;
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
   if (getListAdapter().isEmpty()) {
    menu.findItem(R.id.menu_delete_all).setEnabled(false);
   } else {
    menu.findItem(R.id.menu_delete_all).setEnabled(true);
   }
            return true;
    }



    @Override
    protected void onDestroy() {
            if (alertDialog != null)
                alertDialog.dismiss();
            super.onDestroy();
    }

}
```

Conclusion: We have studied development of Pedometer app for Android OS.

Group B
Assignment No 1

Aim: Write a program to build smart mobile app for user profiling. Objective of this assignment is to develop smart mobile app which can create user profiles based on their preferences so that smart recommendations can be provided at run time.

Tools:
Android SDK,Eclipse, JDK

Theory:

Recommendation systems are widely used on the Internet to assist customers in finding the products or services that best fit with their individual preferences. While current implementations successfully reduce information overload by generating personalized suggestions when searching for objects such as books or movies, recommendation systems so far cannot be
found in another potential field of application: the personalized search for subjects such as applicants in a recruitment scenario. Theory shows that a good match between persons and jobs needs to consider both, the preferences of the recruiter and the preferences of the candidate. Based on this requirement for modeling bilateral selection decisions, we present an approach applying two distinct recommendation systems to the field in order to improve the match between people and jobs. Finally, we present first validation test runs from a
student experiment showing promising results.

Introduction
Personalization systems such as recommender engines in recent years attracted the interest of many researchers and practitioners. Since Resnick and Varian first established the term "recommender system" in 1997 researchers has been improving recommendation quality and scalability of such systems by various means. While some researchers merged content-based with collaborative filtering in order to overcome sparsely problems and combine the advantages of both approaches, others focused on how to reduce the dimensionality of the user-item-matrix underlying collaborative filtering approaches. Today, recommendation systems successfully assist consumers on the Internet in finding products or objects based on items similar to the ones the customer himself previously liked or based on items that other customers similar to him liked in the past. However, personalization systems are not yet applied when searching for subjects. Thus, our research question is: How can the selection of individuals be supported or improved with IS support? We argue that a match between a candidate and a job needs to be bilateral as it requires considering the preferences of the recruiter and the preferences of the candidate.

Program:

```
package com.example.jobseekerapp;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;

public class CreateProfileActivity extends Activity {

	Button btn_search;
	RadioGroup rbJobType, rbSkill, sbCategory, sbLocation;
	RadioButton radiofresher, radioexp, radiofull, radiopart, radioJava,
			radioNet, radioAndroid, radiopune, radiomumabai, radiobang,
			radiohyd, radioPHP;

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

		radiofresher = (RadioButton) findViewById(R.id.radiofresher);
		radioexp = (RadioButton) findViewById(R.id.radioexp);

		radiofull = (RadioButton) findViewById(R.id.radiofull);
		radiopart = (RadioButton) findViewById(R.id.radiopart);

		radioJava = (RadioButton) findViewById(R.id.radioJava);
		radioNet = (RadioButton) findViewById(R.id.radioNet);
		radioAndroid = (RadioButton) findViewById(R.id.radioandroid);
		radioPHP = (RadioButton) findViewById(R.id.radiophp);

		radiopune = (RadioButton) findViewById(R.id.radiopune);
		radiomumabai = (RadioButton) findViewById(R.id.radiomumabai);
		radiobang = (RadioButton) findViewById(R.id.radiobang);
		radiohyd = (RadioButton) findViewById(R.id.radiohyd);

		btn_search = (Button) findViewById(R.id.btn_search);

		btn_search.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {
				// Save Data In Shared Preferences
				saveSharedPreferences();

				Intent intent = new Intent(CreateProfileActivity.this,
						JobsListActivity.class);
```

```
                        startActivity(intent);

                    }
                });

                getDataFromSharedPrefereces();

        }

        private void getDataFromSharedPrefereces() {

                if (getDataLocal(KeysValues.CATEGORY).equals(
                                KeysValues.CATEGORY_FRESHER)) {
                        radiofresher.setChecked(true);
                } else {
                        radioexp.setChecked(true);
                }

                if (getDataLocal(KeysValues.JOBTYPE)
                                .equals(KeysValues.JOBTYPE_FULLTIME)) {
                        radiofull.setChecked(true);
                } else {
                        radiopart.setChecked(true);
                }

                if (getDataLocal(KeysValues.SKILL).equals(KeysValues.SKILL_JAVA)) {
                        radioJava.setChecked(true);
                }
                if (getDataLocal(KeysValues.SKILL).equals(KeysValues.SKILL_NET)) {
                        radioNet.setChecked(true);
                } else if (getDataLocal(KeysValues.SKILL).equals(
                                KeysValues.SKILL_ANDROID)) {
                        radioAndroid.setChecked(true);
                } else {
                        radioPHP.setChecked(true);
                }

                if (getDataLocal(KeysValues.LOCATION).equals(KeysValues.LOCATION_PUNE)) {
                        radiopune.setChecked(true);
                } else if (getDataLocal(KeysValues.LOCATION).equals(
                                KeysValues.LOCATION_MUMBAI)) {
                        radiomumabai.setChecked(true);
                } else if (getDataLocal(KeysValues.LOCATION).equals(
                                KeysValues.LOCATION_HYDRABAD)) {
                        radiohyd.setChecked(true);
                } else {
                        radiobang.setChecked(true);
                }
```

```
	}

	private void saveSharedPreferences() {
		if (radiofresher.isChecked()) {
			saveDataLocal(KeysValues.CATEGORY,
KeysValues.CATEGORY_FRESHER);
		} else {
			saveDataLocal(KeysValues.CATEGORY,
KeysValues.CATEGORY_EXPERIENCE);
		}

		if (radiofull.isChecked()) {
			saveDataLocal(KeysValues.JOBTYPE, KeysValues.JOBTYPE_FULLTIME);
		} else {
			saveDataLocal(KeysValues.JOBTYPE, KeysValues.JOBTYPE_PARTTIME);
		}

		if (radioJava.isChecked()) {
			saveDataLocal(KeysValues.SKILL, KeysValues.SKILL_JAVA);
		} else if (radioNet.isChecked()) {
			saveDataLocal(KeysValues.SKILL, KeysValues.SKILL_NET);
		} else if (radioAndroid.isChecked()) {
			saveDataLocal(KeysValues.SKILL, KeysValues.SKILL_ANDROID);
		} else {
			saveDataLocal(KeysValues.SKILL, KeysValues.SKILL_PHP);
		}

		if (radiopune.isChecked()) {
			saveDataLocal(KeysValues.LOCATION, KeysValues.LOCATION_PUNE);
		} else if (radiomumabai.isChecked()) {
			saveDataLocal(KeysValues.LOCATION, KeysValues.LOCATION_MUMBAI);
		} else if (radiohyd.isChecked()) {
			saveDataLocal(KeysValues.LOCATION,
KeysValues.LOCATION_HYDRABAD);
		} else {
			saveDataLocal(KeysValues.LOCATION,
KeysValues.LOCATION_BANGLORE);
		}

	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.search, menu);
		return true;
	}
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		int id = item.getItemId();
		if (id == R.id.action_add_new_job) {
			Intent intent = new Intent(this, AddNewJobActivity.class);
			startActivity(intent);
		}
		return super.onOptionsItemSelected(item);
	}

	private void saveDataLocal(String key, String value) {
		SharedPreferences preferences = PreferenceManager
				.getDefaultSharedPreferences(this);
		Editor editor = preferences.edit();
		editor.putString(key, value);
		editor.commit();
	}

	private String getDataLocal(String key) {
		SharedPreferences preferences = PreferenceManager
				.getDefaultSharedPreferences(this);
		String value = preferences.getString(key, "");
		return value;
	}

	@Override
	public void onBackPressed() {
		Intent intent = new Intent(CreateProfileActivity.this, JobsListActivity.class);
		startActivity(intent);
		finish();
	}
}
```

Conclusions: Thus, we have studied the Job Seeker application.

Assignment No 2

Aim: To implement MiniMax approach for Tic-Tac-Toe game

Objectives: To study MiniMax approach & to apply it for Tic-Tac-Toe game using Java/Scala/Python-Eclipse.

Tools Requirement:

Hardware: - 64-bit intel i3 processor and above

Software: - 64-bit Fedora OS with in-built eclipse IDE / windows OS with netbeans IDE

Theory:

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.

A simple version of the minimax algorithm, stated below, deals with games such as tic-tac-toe, where each player can win, lose, or draw. If player A can win in one move, his best move is that winning move.

A number of further improvements to the minimax algorithm are possible to deal with certain undesirable situations.

1. **Wait for quiescent states**. It is possible that when a tree node is expanded slightly the state could change drastically. Such information may have a major effect on the choice of which move should be played. To ensure that short-term considerations do not influence our choice of move we might choose to expand the tree until no such major changes occur. This helps to avoid the so-called horizon effect where a very bad move for a player can be temporarily delayed so that it does not appear in the section of the tree that minimax explores (it does not eliminate this possibility, only ameliorate it).

2**. Secondary search**. Another way to tackle the horizon e_ect is to double- check moves before making them. Once minimax has proposed a move itmay be a good idea to expand the search tree a few levels under the node at the bottom of the tree that caused the move to be selected (in order to verify that no nasty surprises are waiting once the move is played).

Program :

```
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Random;

interface Variables
{
   byte P1 = 1, P2 = 2;
   byte Matrix[][] = new byte [3][3];
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   PrintWriter pr = new PrintWriter(System.out, true);
}

class MatrixHandler implements Runnable, Variables
{
   byte x, y;
   byte Count;
   byte xy[][];
   byte Vr[], Vrc;

   MatrixHandler()
   {
      xy = new byte[8][2];
      Vr = new byte[8];
      Vrc = 0;
      for (x=0; x<3; x++)
      {
         for(y=0; y<3; y++)
         {
            Matrix[x][y] = 0;
         }
      }
   }

   boolean Move(byte x, byte y, byte Player)
   {
      this.x = x;
      this.y = y;
      if(Player == P1)
      {
         if(Matrix[x][y] == 0)
            Matrix[x][y] = P1;
         else
            return false;
      }
```

```
Count = 0;
byte i, j;

for(i=0; i<3; i++)
{
    for(j=0; j<3; j++)
    {
        if(Matrix[i][j] == 0)
        {
            xy[Count]  [0] = i;
            xy[Count++][1] = j;
            //System.out.println(PutO(Mat, i, j, (byte)0));
        }
    }
}
Thread T[] = new Thread[Count];
for(i=0; i<Count; i++)
{
    T[i] = new Thread(this, ("Thread No : "+(i)+"  "));
    try
    {
        Thread.sleep(10);
    }
    catch(InterruptedException e)
    {
        pr.println(e);
    }
    T[i].start();
}
try
{
    for(i=0; i<Count; i++)
        T[i].join();
}
catch(InterruptedException e)
{
    pr.println(e);
}
byte Max = Vr[0];
Vr[0] = 0;
for(i=0; i<Count; i++)
{
    if(Max < Vr[i])
    {
        Vr[0] = i;
        Max = Vr[i];
    }
    else if(Max == Vr[i] && xy[i][0]==1 && xy[i][1]==1)
        Vr[0] = i;
        }
        Matrix[xy[Vr[0]][0]][xy[Vr[0]][1]] = P2;
        return true;
    }

    @Override
    public void run()  //To change body of generated methods, choose Tools | Templates.
    {
        Thread t;
        t = Thread.currentThread();
        byte Temp = (byte) t.getName().charAt(t.getName().length()-3);
        Temp -= 48;
        byte Mat[][] = new byte[3][3];

        for(byte i=0; i<3; i++)
            for(byte j=0; j<3; j++)
                Mat[i][j] = Matrix[i][j];

        pr.println(t.getName() + "  " + xy[Temp][0] + "  " + xy[Temp][1]+"\t"+(Vr[Temp]=PutO(Mat,
xy[Temp][0], xy[Temp][1], (byte)0, Temp)));

    }

    public byte PutO(byte Mat[][], byte x, byte y, byte Level, byte Temp)
    {
        byte Val, i, j;
        byte V[][] = new byte[8][3], C=0;
        //Val = 0;

        if((Level%2) == 0)
        {
            Mat[x][y] = P2;
            Val = WonOrNot(Mat, P2);
            //DisplayMatrix(Mat);
            Mat[x][y] = 0;
            Mat[x][y] = P1;
            byte Val1 = WonOrNot(Mat, P1);
            Mat[x][y] = 0;

            if(Val > 0 || Val1 > 0)
            {
                if(Val > Val1)
                    return (byte)(Val-Level);
                else
                    return (byte)(Val1-Level);
            }
        }
        else
        {
```

```
    Mat[x][y] = P1;
    Val = WonOrNot(Mat, P1);
    //DisplayMatrix(Mat);
    Mat[x][y] = 0;
    Mat[x][y] = P2;
    byte Val1 = WonOrNot(Mat, P2);
    Mat[x][y] = 0;

    if(Val > 0 || Val1 > 0)
    {
        if(Val > Val1)
            return (byte)(-Val-Level);
        else
            return (byte)(Val1-Level);
    }
}

for(i=0; i<3; i++)
{
    if((Level%2) == 0)
        Mat[x][y] = P2;
    else
        Mat[x][y] = P1;

    for(j=0; j<3; j++)
    {
        if(Mat[i][j] == 0)
        {
            V[C][0] = i;
            V[C][1] = j;
            V[C++][2] = PutO(Mat, i, j, (byte)(Level+1), Temp);
        }
    }
    Mat[x][y] = 0;
}

if((Level%2) == 0)
{
    for(i=1; i<C; i++)
    {
        if(V[0][2] < V[i][2])
        {
            V[0][0] = V[i][0];
            V[0][1] = V[i][1];
            V[0][2] = V[i][2];
        }
    }
}
else
```

```
    {
        for(i=1; i<C; i++)
        {
            if(V[0][2] > V[i][2])
            {
                V[0][0] = V[i][0];
                V[0][1] = V[i][1];
                V[0][2] = V[i][2];
            }
        }
    }
    Val = (byte)(V[0][2] - Level);
    return Val;
}

public void DisplayMatrix(byte Mat[][])
{
    pr.println("\n\n\n+ - - - - - +");
    for(byte i=0; i<3; i++)
    {
        for(byte j=0; j<3; j++)
        {
            pr.print("| ");
            if(Mat[i][j] == 0)
                pr.print("  ");
            else if(Mat[i][j] == P1)
                pr.print("X ");
            else
                pr.print("O ");
        }
        pr.println("|\n+ - - - - - +");
    }
    pr.println("\n\n");
}

byte MatrixVal(byte Mat[][], byte Player)
{
    byte F=0x10, i, j;

    for(i=0; i<3; i++)
    {
        for(F|=0x0F, j=0; j<3; j++)
        {
            if(Mat[i][j] != Player)
                F &= ~1;
            if(Mat[j][i] != Player)
                F &= ~2;
            if(Mat[j][j] != Player)
                F &= ~4;
            if(Mat[2-j][j] != Player)
                F &= ~8;
            if(Mat[i][j] == 0)
                F &= ~16;
        }
        if((F & 0x0F) != 0)
            return (byte)(F & 0x0F);
    }
    return (byte)(F & 0x10);
}

byte WonOrNot(byte Mat[][], byte Player)
{
    byte V1;
    V1 = MatrixVal(Mat, Player);

    if (V1 == 0x10)
        return 50;
    else if(V1 != 0)
        return 100;
    return 0;
}
}

public class MiniMax extends Applet implements Variables
{
    @Override
    public void init()
    {
        setLayout(new GridLayout(4, 4));
        setFont(new Font("SansSerif", Font.BOLD, 24));
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; j++)
            {
                int k = i * 4 + j;
                if(k > 0)
                    add(new Button("" + k));
            }
        }
    }
    public static void main(String[] args) throws IOException
    {
        MatrixHandler Mobj = new MatrixHandler();
        byte x, y;
        Random R = new Random();
        if ((R.nextInt() % 2) == 1)
        {
            pr.println("\nComputer Plays First !!!\n");
            byte V = (byte) (R.nextInt() % 9);
            if(V < 0)
                V *= -1;
            Matrix[V/3][V%3] = P2;
            Mobj.DisplayMatrix(Matrix);
        }
        else
            pr.println("\nYou Get to Play First !!!\n");
        do
        {
            pr.println("\n\nEnter the Coordinates : ");
            while(br.ready())
                br.read();
            x = (byte)(br.read() - 48);
            while(br.ready())
                br.read();
            y = (byte)(br.read() - 48);
            while(br.ready())
                br.read();
            pr.println("\nEntered Coordinates   : "+x+"  "+y+"\n\n");
            if(x >= 0 && x <3 && y >= 0 && y < 3)
            {
                if(Mobj.Move(x, y, P1))
                {
                    Mobj.DisplayMatrix(Matrix);

                    byte V1 = Mobj.WonOrNot(Matrix, P1);

                    byte V2 = Mobj.WonOrNot(Matrix, P2);
                    if(V1 == 50)
                        pr.println("\nIt's a Draw !!!");
                    else if(V1 == 100)
                        pr.println("\nYou Won !!!");
                    else if(V2 == 100)
                        pr.println("\nComputer Won !!!");
                }
            }
        } while(Mobj.WonOrNot(Matrix, P1) == 0 && Mobj.WonOrNot(Matrix, P2) == 0);
    }
}
```

Case study (Any example) :

The following example of a zero-sum game, where **A** and **B** make simultaneous moves, illustrates *minimax* solutions. Suppose each player has three choices and consider the payoff matrix for **A** displayed at right. Assume the payoff matrix for **B** is the same matrix with the signs reversed (i.e. if the choices are A1 and B1 then **B** pays 3 to **A**). Then, the minimax choice for **A** is A2 since the worst possible result is then having to pay 1, while the simple minimax choice for **B** is B2 since the worst possible result is then no payment. However, this solution is not stable, since if **B** believes **A** will choose A2 then **B** will choose B1 to gain 1; then if **A** believes **B** will choose B1 then **A** will choose A1 to gain 3; and then **B** will choose B2; and eventually both players will realize the difficulty of making a choice. So a more stable strategy is needed.

	B chooses B1	**B chooses B2**	**B chooses B3**
A chooses A1	+3	−2	+2
A chooses A2	−1	0	+4
A chooses A3	−4	−3	+1

Some choices are *dominated* by others and can be eliminated: **A** will not choose A3 since either A1 or A2 will produce a better result, no matter what **B** chooses; **B** will not choose B3 since some mixtures of B1 and B2 will produce a better result, no matter what **A** chooses.

A can avoid having to make an expected payment of more than 1⁄3 by choosing A1 with probability 1⁄6 and A2 with probability 5⁄6: The expected payoff for **A** would be 3 × (1⁄6) − 1 × (5⁄6) = −1⁄3 in case **B** chose B1 and −2 × (1⁄6) + 0 × (5⁄6) = −1⁄3 in case **B** chose B2. Similarly, **B** can ensure an expected gain of at least 1/3, no matter what **A** chooses, by using a randomized strategy of choosing B1 with probability 1⁄3 and B2 with probability 2⁄3. These mixed minimax strategies are now stable and cannot be improved.

Mathematical modeling:

In classical statistical decision theory, we have an estimator δ that is used to estimate a parameter $\theta \in \Theta$. We also assume a risk function $R(\theta, \delta)$, usually specified as the integral of a loss function. In this framework, $\tilde{\delta}$ is called **minimax** if it satisfies

$$\sup_\theta R(\theta, \tilde{\delta}) = \inf_\delta \sup_\theta R(\theta, \delta).$$

An alternative criterion in the decision theoretic framework is the Bayes estimator in the presence of a prior distribution Π. An estimator is Bayes if it minimizes the *average* risk

$$\int_\Theta R(\theta, \delta)\, d\Pi(\theta).$$

Optimized Code:

```
function minimax(node, depth, maximizingPlayer)
    if depth = 0 or node is a terminal node
        return the heuristic value of node
    if maximizingPlayer
        bestValue := -∞
        for each child of node
            val := minimax(child, depth - 1, FALSE)
            bestValue := max(bestValue, val)
        return bestValue
    else
        bestValue := +∞
        for each child of node
            val := minimax(child, depth - 1, TRUE)
            bestValue := min(bestValue, val)
        return bestValue

(* Initial call for maximizing player *)
minimax(origin, depth, TRUE)
```

With alpha-Beta pruning:

```
function MINIMAX-AB(N, A, B) is
  if N is deep enough then
    return the estimated score of this leaf
  else
    alpha = a; beta = b;
    if N is a Min node then
       For each successor Ni of N
         beta = min{beta, MINIMAX-AB(Ni, alpha, beta)}
         if alpha >= beta then return alpha
       end for
       return beta
    else
       For each successor Ni of N
         alpha = max{alpha, MINIMAX-AB(Ni, alpha, beta)}
         if alpha >= beta then return beta
       end for
       return alpha
    end if
  end if
end function
```

Code complexity: O(n)

Conclusions: Studied and implemented MiniMax approach for Tic-Tac-Toe game.

Assignment No 3

Aim: Implementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using XML/JSON use C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities.

Objectives: To understand the uninformed search for space search and efficient routing of graphs.

Tools Requirement:

Hardware: - 500 MB Hard Disk, 512 MB RAM.

Software: - JDK 6 or above, gcc-c++, Python 2.x

Theory:

Uninformed Search
The uninformed search methods offer a variety of techniques for graph search, each with its own advantages and disadvantages. These methods are explored here with a discussion of their characteristics and complexities.
It varies according to the state and distance in space search.If we want to deal with long distances, we can use the depth limited search. If we want to apply search for the short paths, breadth first search is useful to make the traversal as it is going to make search comparatively faster.
Big-O notation will be used to compare the algorithms. This notation defines the asymptotic upper bound of the algorithm given the depth (*d*) of the tree and the branching factor, or the average number of branches (*b*) from each node. There are a number of common complexities that exist for search algorithms. A search algorithm is characterized as exhaustive when it can search every node in the graph in search of the goal. If the goal is not present in the graph, the algorithm will terminate, but will search each and every node in a systematic way.

Common orders of search functions.
O-Notation Order
O(1) Constant (regardless of the number of nodes)
O(n) Linear (consistent with the number of nodes)
O(log n) Logarithmic
O(n^2) Quadratic
O(c^n) Geometric
O(n!) Combinatorial

Big-O notation provides a worst-case measure of the complexity of a search algorithm and is a common comparison tool for algorithms. We’ll compare the search algorithms using *space complexity* (a measure of the memory required during the search) and *time complexity* (worst-case time required to find a solution). We’ll also review the algorithm for *completeness* (can the algorithm find a path to a goal node if it’s present in the graph) and *optimality* (finds the lowest cost solution available).

Depth-First Search (DFS)

The Depth-First Search (DFS) algorithm is a technique for searching a graph that begins at the root node, and exhaustively searches each branch to its greatest depth before backtracking to previously unexplored branches. Nodes found but yet to be reviewed are stored in a LIFO queue (also known as a *stack*).

The space complexity for DFS is O(b^d) where the time complexity is geometric (O(b^d)). This can be very problematic on deep branching graphs, as the algorithm will continue to the maximum depth of the graph. If loops are present in the graph, then DFS will follow these cycles indefinitely. For this reason, the DFS algorithm is not complete, as cycles can prohibit the algorithm from finding the goal. If cycles are not present in the graph, then the algorithm is complete (will always find the goal node). The DFS algorithm is also not optimal, but can be made optimal using path checking (to ensure the shortest path to the goal is found).

Graph algorithms can be implemented either recursively or using a stack to maintain the list of nodes that must be enumerated.

Depth-Limited Search (DLS)

Depth-Limited Search (DLS) is a modification of depth-first search that minimizes the depth that the search algorithm may go. In addition to starting with a root and goal node, a depth is provided that the algorithm will not descend below. Any nodes below that depth are omitted from the search. This modification keeps the algorithm from indefinitely cycling by halting the search after the pre-imposed depth.

While the algorithm does remove the possibility of infinitely looping in the graph, it also reduces the scope of the search. If the goal node had been one of the nodes marked 'X', it would not have been found, making the search algorithm incomplete. The algorithm can be complete if the search depth is that of the tree itself (in this case *d* is three). The technique is also not optimal since the first path may be found to the goal instead of the shortest path. The time and space complexity of depth-limited search is similar to DFS, from which this algorithm is derived. Space complexity is O(b^d) and time complexity is O(b^d), but *d* in this case is the imposed depth of the search and not the maximum depth of the graph.

Iterative Deepening Search (IDS)

Iterative Deepening Search (IDS) is a derivative of DLS and combines the features of depth-first search with that of breadth-first search. IDS operates by performing DLS searches with increased depths until the goal is found. The depth begins at one, and increases until the goal is found, or no further nodes can be enumerated.

IDS combines depth-first search with breadth first search. By minimizing the depth of the search, we force the algorithm to also search the breadth of the graph. If the goal is not found, the depth that the algorithm is permitted to search is increased and the algorithm is started again. IDS is advantageous because it's not susceptible to cycles (a characteristic of DLS, upon which it's based). It also finds the goal nearest to the root node, as does the BFS algorithm (which will be detailed next). For this reason, it's a preferred algorithm when the depth of the solution is not known. The time complexity for IDS is identical to that of DFS and DLS, O(b^d).

Space complexity of IDS is O(b^d). Unlike DFS and DLS, IDS is will always find the best solution and therefore, it is both complete and optimal.

Breadth-First Search (BFS)

In Breadth-First Search (BFS), we search the graph from the root node inorder of the distance from the root. Because the order search is nearest the root, BFS is guaranteed to find the best possible solution (shallowest) in a non-weighted graph, and is therefore also complete. Rather than digging deep down into the graph, progressing further and further from the root (as is the case with DFS), BFS checks each node nearest the root before descending to the next level.
The implementation of BFS uses a FIFO (first-in-first-out) queue, differing from the stack (LIFO) implementation for DFS. As new nodes are found to be searched, these nodes are checked against the goal, and if the goal is not found, the new nodes are added to the queue. To continue the search, the oldest node is dequeued (FIFO order). Using FIFO order for new node search, we always check the oldest nodes first, resulting in breadth-first review.

The disadvantage of BFS is that each node that is searched is required to be stored (space complexity is $O(b^d)$). The entire depth of the tree does not have to be searched, so *d* in this context is the depth of the solution, and
not the maximum depth of the tree. Time complexity is also $O(b^d)$.
In practical implementations of BFS, and other search algorithms, a closed list is maintained that contains those nodes in the graph that have been visited. This allows the algorithm to efficiently search the graph without re-visiting nodes. In implementations where the graph is weighted, keeping a closed list is not possible.

Bidirectional Search
The Bidirectional Search algorithm is a derivative of BFS that operates by performing two breadth-first searches simultaneously, one beginning from the root node and the other from the goal node. When the two searches meet in the middle, a path can be reconstructed from the root to the goal. The searches meeting is determined when a common node is found. This is accomplished by keeping a closed list of the nodes visited.
Bidirectional search is an interesting idea, but requires that we know the goal that we're seeking in the graph. This isn't always practical, which limits the application of the algorithm. When it can be determined, the algorithm has useful characteristics. The time and space complexity for bidirectional search is O(bd/2), since we're only required to search half of the depth of the tree. Since it is based on BFS, bidirectional search is both complete and optimal.

Uniform-Cost Search (UCS)
One advantage of BFS is that it always finds the shallowest solution. Butconsider the edge having a cost associated with it. The shallowest solution may not be the best, and a deeper solution with a reduced path cost would be better. Uniform -Cost Search (UCS) can be applied to find the least-cost path through a graph by maintaining an ordered list of nodes in order of descending cost. This allows us to evaluate the least cost path first
Uniform-cost search is an uninformed search method because no heuristic is actually used. The algorithm measures the actual cost of the path without attempting to estimate it.

The algorithm for UCS uses the accumulated path cost and a priority queue to determine the path to evaluate. The priority queue (sorted from least cost to greatest) contains the nodes to be evaluated. As node children are evaluated, we add their cost to the node with the aggregate sum of the current path. This node is then added to the queue, and when all children have been evaluated, the queue is sorted in order of ascending cost. When the first element in the priority queue is the goal node, then the best solution has been found.

The UCS algorithm is easily demonstrated using our example graph shows the state of the priority queue as the nodes are evaluated. At step one, the initial node has been added to the priority queue, with a cost of zero. At step two, each of the three connected nodes are evaluated and added to the priority queue. When no further children are available to evaluate, the priority queue is sorted to place them in ascending cost order.

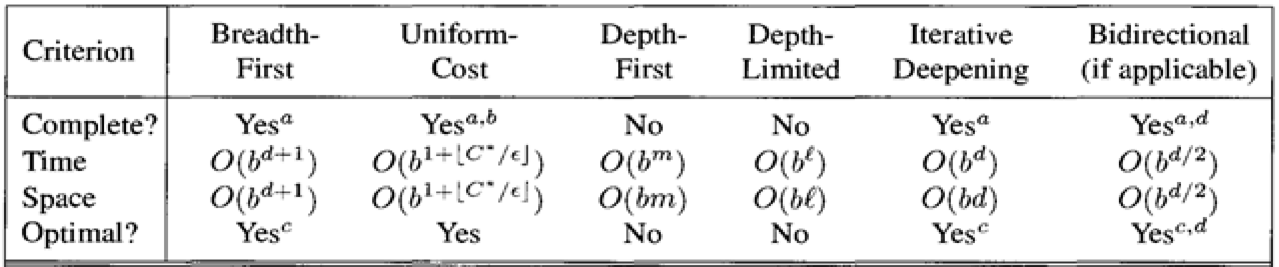
UCS is optimal and can be complete, but only if the edge costs are non-negative (the summed path cost always increases). Time and space complexity are the same as BFS, O(bd) for each, as it's possible for the entire tree to be evaluated.

Program with proper indentation& comments:

<u>//Depth First Search</u>

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
 int i;
 reach[v]=1;
 for(i=1;i<=n;i++)
  if(a[v][i] && !reach[i])
  {
   printf("\n %d->%d",v,i);
   dfs(i);
  }
}
void main()
{
 int i,j,count=0;
 clrscr();
 printf("\n Enter number of vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  reach[i]=0;
  for(j=1;j<=n;j++)
   a[i][j]=0;
 }
 printf("\n Enter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
   scanf("%d",&a[i][j]);
 dfs(1);
 printf("\n");
 for(i=1;i<=n;i++)
 {
  if(reach[i])
   count++;
 }
 if(count==n)
  printf("\n Graph is connected");
 else
  printf("\n Graph is not connected");
 getch();
//Depth Limited Search
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.Stack;

public class DepthLimitedSearch {

    private Stack<Integer> stack;
    private int numberOfNodes;
    private static final int MAX_DEPTH = 3;

    public DepthLimitedSearch(int numberOfNodes) {
        this.numberOfNodes = numberOfNodes;
        this.stack = new Stack<Integer>();
    }

    public void depthLimitedSearch(int adjacencyMatrix[][], int source) {
        int visited[] = new int[numberOfNodes + 1];
        int element, destination;
        int depth = 0;

        System.out.println(source + " at depth " + depth);
        stack.push(source);
        visited[source] = 1;
        depth = 0;

        while (!stack.isEmpty()) {
            element = stack.peek();
            destination = element;
            while (destination <= numberOfNodes) {
                if (depth < MAX_DEPTH) {
                    if (adjacencyMatrix[element][destination] == 1 && visited[destination] == 0) {
                        stack.push(destination);
                        visited[destination] = 1;
                        depth++;
                        System.out.println(destination + " at depth " + depth);
                        element = destination;
                        destination = 1;
                    }
                } else {
                    return;
                }
                destination++;
            }
            stack.pop();
            depth--;
        }
    }

    public static void main(String[] args) {
        int number_of_nodes, source;
        Scanner scanner = null;
        try {
            System.out.println("Enter the number of nodes in the graph");
            scanner = new Scanner(System.in);
            number_of_nodes = scanner.nextInt();

            int adjacency_matrix[][] = new int[number_of_nodes + 1][number_of_nodes + 1];
            System.out.println("Enter the adjacency matrix");
            for (int i = 1; i <= number_of_nodes; i++) {
                for (int j = 1; j <= number_of_nodes; j++) {
                    adjacency_matrix[i][j] = scanner.nextInt();
                }
            }

            System.out.println("Enter the source for the graph");
            source = scanner.nextInt();

            System.out.println("The Depth limited Search Traversal of Max Depth 3 is");
            DepthLimitedSearch depthLimitedSearch = new DepthLimitedSearch(number_of_nodes);
            depthLimitedSearch.depthLimitedSearch(adjacency_matrix, source);

        } catch (InputMismatchException inputMismatch) {
            System.out.println("Wrong Input format");
        }
        scanner.close();
    }
}
```

Code complexity:



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.17 Evaluation of search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Conclusions: In this way, we have learnt aboutImplementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities.

Assignment No 4

Aim: To develop app for Android OS regarding Book recommender system.

Tools:
Eclipse, Android SDK, Java.

Theory:

Virtually every student has had an online experience where a website makes personalized recommendations in hopes of future sales or ongoing traffic. Amazon.com tells you "Customers Who Bought This Item Also Bought", YouTube makes suggestions for other videos to watch, and NetFlix ran a contest with a million dollar prize for an algorithm that would improve their movie recommendations. In this assignment, students write a program that makes personalized book recommendations using algorithms with increasing levels of sophistication.

After reading a pre-collected set of ratings for a list of books, the program makes recommendations for a particular reader based on a small set of sample ratings from that reader and the preferences of other readers in the community. The assignment was inspired by machine learning research used to predict book preferences for Chapters.Indigo.ca. It provides the opportunity to discuss similarity measures for non-trivial objects and alludes to machine learning techniques.

Meta Information

Summary	Read book ratings for a set of users from a file. Define a similarity measure for any pair of readers. Based on reader-similarity, use ratings from the reader community to recommend new books for a particular user.
Topics	arrays, reading from files. It requires at least a one-dimensional array of Strings and either a 2D array of integers or a 1D array of objects. It could also be designed to use dictionaries and/or include a GUI.
Audience	CS1. It has also been used in senior high-school classes.
Difficulty	This is an intermediate assignment, taking 2 or 3 weeks for a CS1 student. It can be made more or less complicated by allowing the students varying degrees of flexibility in defining their reader-similarity measure and data structures.
Strengths	• Students are inspired by the fact that the Netflix contest offered significant money for the real-life version of this problem. They like to see their programs make personalized predictions for themselves and their friends. • Instructors like the fact that it can be assigned when students haven't learned about more complex data-structures and that it can be a starting point for talking about other CS ideas: weighted averages, comparing non-trivial objects, designing distance measures between objects and writing comparators, sparse data and machine learning. • Instructors also like that is very flexible allowing them to meet other curriculum expectations as needed.
Weaknesses	• If the rating scheme and distance measure (between two readers) are left up to

	the student, they may design something that doesn't make good predictions. Without direction, students tend to design 1-5 or 0-5 based scheme similar to giving a book "3 stars". This makes generating a similarity measure more complicated. If you are giving the students a file of pre-collected ratings, you have to give them the rating scheme. • In order to work, the assignment needs real data. You can use the book list and associated ratings files that I've collected or you can recruit a community of people to rate a large set of items. You can't generate the input data randomly. • The Netflix contest is still reasonably current, but it will soon be old news. That won't make the assignment less doable, but it won't be as nifty as it becomes less current.
Dependencies	No external libraries required.
Variants	• Instead of using the book data that I've provided here, some teachers had students collect their own data set so that the predictions could be about something other than books. It can be a lot of work to get enough data but this can also be a very cool thing to do with a large class. You can encourage each student in the class to submit their ratings with a simple web form. • The amount of freedom given to the students to design the rating scheme and the similarity function (between pairs of readers) can be varied. See the cautions in the weaknesses section. • Including a GUI component works well, but the assignment is also fine with only a text-based interface. • While the prediction algorithms can be implemented using a 2D array of integer ratings, many of these ratings are zero leading to a sparse matrix. One way to make the assignment harder is to require the use of a more sophisticated data-structure to store the ratings data without explicitly storing all the zeros. This can drive home the idea of having different representations for the same data. • Reading the input file can be made slightly more difficult by combining the book list data and the ratings data into a single input file that must be parsed by the program.

Data Set

The data set focuses on this audience. I worked with a community librarian to identify 55 books that in some sense covered the spectrum of books read by typical high school seniors in Canada. I then recruited 86 readers to rate the books in the set using the following rating scheme:

Rating	**Meaning**

-5	Hated it!
-3	Didn't like it
0	Haven't read it
1	ok - neither hot nor cold about it
3	Liked it!
5	Really liked it!

Here are the resulting data files.

- The list of 55 books
- Ratings file for the above book list (from 86 different readers)

Solution Code

I have solutions to the assignment in both Java and Python and am happy to distribute them to instructors who are considering using the project. I ask that no solutions be posted on the web.

Example Handouts

Diane Horton gave this assignment at the University of Toronto in CS1 and we are aware that it was given in various local high schools. The CS1 handout below is almost exactly what was used with the omission of instructions about our local submission and marking mechanisms. The high school handouts are templates that were customized by different teachers. Teachers were intended to take material from the teacher supplement handout and paste it into the student handout template as needed.

- HTML handout used Winter 2010 in CS1
- Highschool Student handout Template (MS Word)
- Teacher Supplement (MS Word)

Program:

```
package com.example.bookrecommender;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.ListView;
import android.widget.Spinner;

public class MainActivity extends Activity {
        private Spinner searchView;
        private ArrayAdapter<String> searchAdapter, listAdapter;
        private ListView listView;
        private ArrayList<String> dataList;
        private DatabaseHelper databaseHelper;
        private BookListAdapter bookListAdapter;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                // initialize UI components
                searchView = (Spinner) findViewById(R.id.searchView);
                listView = (ListView) findViewById(R.id.list);
                searchAdapter = new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_1,
getResources().getStringArray(R.array.searchCategory));

                searchView.setAdapter(searchAdapter);



                databaseHelper = new DatabaseHelper(getApplicationContext());
                ArrayList<BookBean> list =
databaseHelper.getBookList(searchView.getSelectedItem().toString());
                bookListAdapter = new BookListAdapter(getApplicationContext(), list);
                listView.setAdapter(bookListAdapter);



        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
                int id = item.getItemId();
                if (id == R.id.action_add_book) {
                        Intent intent=new Intent(this,AddNewBookActivity.class);
                        startActivity(intent);
                        finish();
        }else if(id == R.id.action_rate_book){
            Intent intent=new Intent(this,BookRatingActivity.class);
            startActivity(intent);
            finish();
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * Find book as per search text
     *
     * @param bookType
     */
    private void findBookList(String bookType) {
        String[] booklist = getBooklistfromResources(bookType);
        listAdapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, booklist);
        listView.setAdapter(listAdapter);
    }

    /**
     * get list of books from resources
     *
     * @param bookType
     * @return
     */
    private String[] getBooklistfromResources(String bookType) {
        String[] booklist = null;

        if (bookType.equals(getString(R.string.novel))) {
            booklist = getResources().getStringArray(R.array.novel);
        } else if (bookType.equals(getString(R.string.fantasy))) {
            booklist = getResources().getStringArray(R.array.fantasy);
        } else if (bookType.equals(getString(R.string.literature))) {
            booklist = getResources().getStringArray(R.array.literature);
        } else if (bookType.equals(getString(R.string.scienceFiction))) {
            booklist = getResources().getStringArray(R.array.scienceFiction);
        } else if (bookType.equals(getString(R.string.java))) {
            booklist = getResources().getStringArray(R.array.java);
        }
        return booklist;

    }

}
```

Conclusions: In this way we have studied the Book Recommender System.

Assignment No 5

Aim: To implement K means algorithm for clustering of given data

Objectives: To validate the K means algorithm for classification.
To observe the pros and cons of the algorithm

Assignment Statement:
Implement k-means algorithm for clustering.

Data of children belonging to different age groups.
Perform some specific activities. Formulate the feature vector for following parameters:

1.Height
2.Weight
3.Age
4.IQ

Formulate the data for 40 children to form 3 clusters.

Relevant theory and concept:

What is clustering?
Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.
Types of clustering:
Hierarchical algorithms: these find successive clusters using previously established clusters.
1.Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
3. Partitional clustering: Partitional algorithms determine all clusters at once. They include:

K-means and derivatives
Fuzzy c-means clustering
QT clustering algorithm

Common Distance measures:
Distance measure will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.
They include:
1)The Euclidean distance (also called 2-norm distance) is given by:

$$d(x, y) = \sum_{i=1}^{p} | x_i - y_i |$$

2) The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x,y) = \sqrt[2]{\sum_{i=1}^{p} | x_i - y_i |^2}$$

K-MEANS CLUSTERING:

Objects are classified into No of groups, so that they are as much dissimilar as possible from one group to another group but as much similar as possible within each group.

The k-means algorithm is an algorithm to cluster*n* objects based on attributes into *k*partitions, where $k<n$.

It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.

It assumes that the object attributes form a vector space.

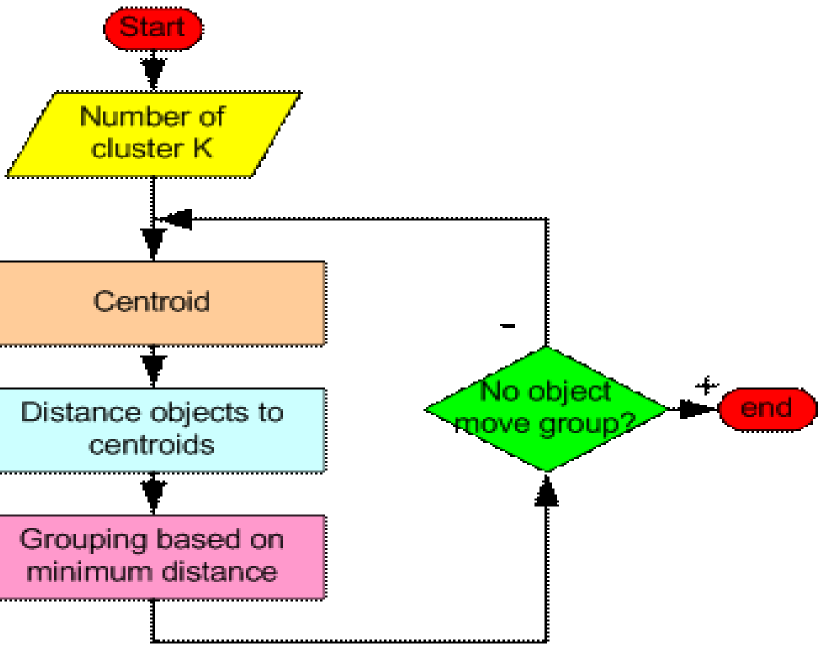
Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.

K is positive integer number.

The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

where x_n is a vector representing the the nth data point and u_j is the geometric centroid of the data points in S_j.


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

The k-means algorithm is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k<n$.

Step 1: Begin with a decision on the value of k =number of clusters. (Here k=3)

Step 2: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:

1.Take the first k training sample as single-element clusters

2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

Step 3: Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that

cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
Step 4 . Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

Numerical Example of K-Means Clustering

We have 4 medicines as our training data points object and each medicine has 2 attributes. Each attribute represents coordinate of the object. We have to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

Object	Attribute 1 (X)	Attribute 2 (Y)
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Step 1:
Initial value of centroids*:* Suppose we use medicine A and medicine B as the first centroids.
Let and c_1 and c_2 denote the coordinate of the centroids, then c_1=(1,1) and c_2=(2,1).

Step 2:
Objects-Centroids distance*:* we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 is

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{array}{ll} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{array}$$

$$\begin{array}{cccc} A & B & C & D \end{array}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{array}{c} X \\ Y \end{array}$$

Each column in the distance matrix symbolizes the object.
The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid.

For example, distance from medicine C = (4, 3) to the first centroid $\mathbf{c}_1 = (1,1)$ is ,
$\sqrt{(4-1)^2+(3-1)^2} = 3.61$ and its distance to the second centroid is $\mathbf{c}_2 = (2,1)$ is
$\sqrt{(4-2)^2+(3-1)^2} = 2.83$

Iteration-1, Objects-Centroids distances:The next step is to compute the distance of all objects to the new centroids.
Similar to step 2, we have distance matrix at iteration 1 is

$$\mathbf{D}^1 = \begin{array}{c} \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \begin{array}{l} \mathbf{c}_1 = (1,1) \quad group-1 \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) \quad group-2 \end{array} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{array}{c} X \\ Y \end{array}$$

<u>Iteration-1, Objects clustering:</u> Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain. The Group matrix is shown below

$$\mathbf{G}^1 = \begin{array}{c} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{array}{l} group-1 \\ group-2 \end{array} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array}$$

<u>Iteration 2, determine centroids:</u>Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are $\mathbf{c}_1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1\frac{1}{2}, 1)$ and, $\mathbf{c}_2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4\frac{1}{2}, 3\frac{1}{2})$

<u>Iteration-2, Objects-Centroids distances</u>: Repeat step 2 again, we have new distance matrix at iteration 2 as

$$\mathbf{D}^2 = \begin{array}{c} \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \begin{array}{l} \mathbf{c}_1 = (1\frac{1}{2}, 1) \quad group-1 \\ \mathbf{c}_2 = (4\frac{1}{2}, 3\frac{1}{2}) \quad group-2 \end{array} \\ \begin{array}{cccc} A & B & C & D \end{array} \end{array}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{array}{c} X \\ Y \end{array}$$

<u>Iteration-2, Objects clustering:</u>Again, we assign each object based on the minimum distance.

$$\mathbf{G}^2 = \begin{array}{cc} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} & \begin{array}{l} group-1 \\ group-2 \end{array} \\ \begin{array}{cccc} A & B & C & D \end{array} & \end{array}$$

Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore. Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed.

We get the final grouping as the results as:

Object	Feature1(X): weight index	Feature2 (Y): pH	Group (result)
Medicine A	1	1	1
Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2

Applications of K-Mean Clustering

k-means clustering can be applied to machine learning or data mining

Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization or Image Segmentation).

Also used for choosing color palettes on old-fashioned graphical display devices and Image Quantization.

Implementation (source Code):

```
/*Implement k-means algorithm for clustering.
Data of children belonging to different age groups.
Perform some specific activities. Formulate the feature vector for following parameters:
        1.Height
        2.Weight
        3.Age
        4.IQ
Formulate the data for 40 children to form 3 clusters.
*/

#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>

#define Random
#define Testing
```

```
#define k       3
#define Ht      0
#define Wt      1
#define Iq      3
#define Age     2

using namespace std;

class Children
{
        int C;

        int Final[k][40];
        int NFinal[k][40];
        float Data[4][40];
        double Centroid[4][k];
        double Distance[k][40];

  public:

          Children();

        int Ins(float H, int W, int A, int I)
        {
                if (C >= 40)
                        return 1;
                Data[Ht][C] = H, Data[Wt][C] = W, Data[Age][C] = A, Data[Iq][C] = I;
                C++;
                return 0;
        }
        void Show();
        void Cluster();
} Child;

Children::Children()
{
        C = 0;
        for (int i = 0; i < 4; i++)
        {

                for (int j = 0; j < 40; j++)
                {
                        Data[i][j] = 0;
                        if(i<k)
                                NFinal[i][j] = Final[i][j]=0;
                }
        }
}
```

```
void Children::Show()
{
	cout << "\n\n";
	cout << "Children  Height (Ft)\tWeight (Kg)\tAge\t   Iq ";
	for (int i = 0; i < C; i++)
		cout << "\n " << (i + 1) << "\t    " << Data[Ht][i] << "\t\t   " << Data[Wt][i] << "\t\t" <<
Data[Age][i] << "\t   " << Data[Iq][i];
}

void Children::Cluster()
{
	int Index[40], i, j;
	//srand(time(0));
	for (i = 0; i < k; i++)
	{
	  Label:
		int tmp = rand() % C;
		for (j = 0; j < i; j++)
		{
			if (Index[j] == tmp)
				goto Label;
		}
		Index[i] = tmp;
	}
	for (i = 0; i < (k - 1); i++)
	{
		for (j = 0; j < (k - 1 - i); j++)
		{
			if (Index[j] > Index[j + 1])
			{
				int tmp = Index[j];
				Index[j] = Index[j + 1];
				Index[j + 1] = tmp;
			}
		}
	}

	#ifdef Testing
		cout << "\n\n\nIndices Choosen : ";

		#ifndef Random
			for(i=0; i<k; i++)
				Index[i] = i;
		#endif

		for(i=0; i<k; i++)
			cout << Index[i] << "  ";
	#endif
	for(i=0;i<4;i++)
		for(j=0;j<k;j++)
			Centroid[i][j] = Data[i][Index[j]];

RmsCalculation:

	for (i = 0; i < k; i++)
	{
		double Rms;
		for (int m = 0, n = 0; m < C; m++)
		{
			Rms = 0;
			for (n = 0; n < 4; n++)
				if(Centroid[n][i])
					Rms += pow((Data[n][m] - Centroid[n][i]), 2);
			if(Centroid[0][i])
				Distance[i][m] = sqrt(Rms);
		}
	}

	for (i = 0; i < C; i++)
	{
		int x = i, y = 0;

		for (j = 1; j < k; j++)
			if (Distance[j][i] < Distance[y][x])
				x = i, y = j, NFinal[j][i] = 0;
		NFinal[y][x] = 1;
	}

#ifdef Testing
	cout<<"\n\n\nDistance Matrix...\n\n";
	for (i = 0; i < C; i++)
	{
		cout << "\n\n";
		for (j = 0; j < k; j++)
			//cout << Distance[j][i] << "  ";
			printf("%-12f",Distance[j][i]);
	}
	cout << "\n\n\nFinal Matrix...\n\n";
	for (i = 0; i < C; i++)
	{
		cout << "\n\n";
		for (j = 0; j < k; j++)
			printf("%-4d",NFinal[j][i]);
	}
	cout<<"\n\n";
#endif
```

```
bool Match=true;

for(i=0; i<k; i++)
        for(j=0; j<C; j++)
                if(NFinal[i][j] != Final[i][j])
                        Match = false;

if(Match == false)
{
        for(i=0; i<k; i++)
                for(j=0; j<C; j++)
                        Final[i][j] = NFinal[i][j];

        for(i=0;i<k;i++)
        {
                int Count = -1,m;
                for(j=0;j<C;j++)
                {
                        if(NFinal[i][j])
                                Count++;
                }
                if(Count)
                {
                        for(j=0;j<4;j++)
                        {
                                for(m=0,Centroid[j][i]=0;m<C;m++)
                                        Centroid[j][i] += NFinal[i][m]*Data[j][m];
                                #ifdef Testing
                                        Centroid[j][i]/= (Count+1);
                                #endif
                        }
                }
                else
                        for(j=0;j<4;j++)
                                Centroid[j][i] = 0;
        }

        #ifdef Testing
                cout << "\n\n\nCentroid Matrix...\n\n";
                for (i = 0; i < 4; i++)
                {
                        cout << "\n\n";
                        for (j = 0; j < k; j++)
                                printf("%-12f",Centroid[i][j]);
                }
                cout<<"\n\n";
        #endif
        goto RmsCalculation;
}
```

```
        else
        {
                for(i=0; i<k; i++)
                {
                        cout<<"\n\n\n\nGroup "<<(char)(65+i)<<" : \n";
                        cout << "\n\n";
                        cout << "Children  Height (Ft)\tWeight (Kg)\tAge\t   Iq \n";
                        for(j=0; j<C; j++)
                        {
                                if(NFinal[i][j])
cout << "\n "<< (j + 1) << "\t    " << Data[Ht][j] << "\t\t   " << Data[Wt][j] << "\t\t" << Data[Age][j]
<< "\t   " << Data[Iq][j];  }
                }
        }
}

int main()
{
        Child.Ins(5.6, 70, 15, 50);
        Child.Ins(5.8, 58, 16, 50);
        Child.Ins(5.1, 65, 17, 65);
        Child.Ins(5.3, 68, 17, 70);
        Child.Ins(5.9, 69, 16, 45);
        Child.Ins(6.0, 72, 15, 30);
        Child.Ins(6.2, 78, 16, 90);
        Child.Ins(5.2, 90, 16, 85);
        Child.Ins(5.2, 84, 16, 70);
        Child.Ins(6.0, 86, 16, 80);
        Child.Show();
        Child.Cluster();
        cout << "\n";
        return 0;
```

<u>Testing:</u>

```
Last login:Prashant Tue Jul 28 06:15:15 on ttys000
Password:
sh-3.2# g++ MEAN.CPP
sh-3.2# ./a.out

Children  Height (Ft)  Weight (Kg)    Age       Iq
 1          5.6           70           15        50
 2          5.8           58           16        50
 3          5.1           65           17        65
 4          5.3           68           17        70
 5          5.9           69           16        45
 6          6             72           15        30
 7          6.2           78           16        90
 8          5.2             90            16         85
 9          5.2             84            16         70
 10         6               86            16         80


Indices Choqsen : 0  1  2


Distance Matrix...
0.000000    12.043255   15.945219
12.043255   0.000000    16.597891
15.945219   16.597891   0.000000
20.201238   22.388613   5.834381
5.204805    12.083460   20.436242
20.103731   24.434402   35.760453
40.808823   44.723148   28.217193
40.325674   47.427418   32.031391
24.436857   32.807926   19.672570
34.017055   41.037056   25.842020

Final Matrix...

1   0   0
0   1   0
0   0   1
0   0   1
1   0   0
1   0   0
0   0   1
0   0   1
0   0   1
0   0   1

Centroid Matrix...

5.833333    0.000000    5.500000
70.333333   0.000000    78.500000
15.333333   0.000000    16.333333
41.666667   0.000000    76.666667



Distance Matrix...
 8.349917    12.043255   28.020508

14.899702   0.000000    33.638701

24.004259   16.597891   17.859607
```

```
28.483055   22.388613   12.457082

3.652092    12.083460   33.065071

11.791004   24.434402   47.138684

48.943516   44.723148   13.365212

47.596230   47.427418   14.208996

31.470639   32.807926   8.654222

41.416918   41.037056   8.229351

Final Matrix...

1   0   0
0   1   0
0   1   1
0   0   1
1   0   0
1   0   0
0   0   1
0   0   1
0   0   1
0   0   1

Centroid Matrix...

5.833333    5.450000    5.500000
70.333333   61.500000   78.500000
15.333333   16.500000   16.333333
41.666667   57.500000   76.666667

Distance Matrix...

8.349917    11.435580   28.020508
14.899702   8.298946    33.638701
24.004259   8.298946    17.859607
28.483055   14.098670   12.457082
3.652092    14.592892   33.065071
11.791004   29.479696   47.138684

48.943516   36.459738   13.365212
47.596230   39.608238   14.208996
31.470639   25.745145   8.654222
41.416918   33.272398   8.229351

Final Matrix...
1   0   0
0   1   0
```

```
0   1   1
0   0   1
1   0   0
1   0   0
0   0   1
0   0   1
0   0   1
0   0   1
Group A :

Children  Height (Ft)  Weight (Kg)   Age      Iq
 1        5.6            70          15       50
 5        5.9            69          16       45
 6        6              72          15       30

Group B :

Children  Height (Ft)  Weight (Kg)   Age      Iq
 2        5.8            58          16       50
 3        5.1            65          17       65

Group C:
Children  Height (Ft)  Weight (Kg)   Age      Iq
 3        5.1            65          17       65
 4        5.3            68          17       70
 7        6.2            78          16       90
 8        5.2            90          16       85
 9        5.2            84          16       70
 10       6              86          16       80
sh-3.2#
```

Code Complexity: Regarding computational complexity, finding the optimal solution to the *k*-means clustering problem for observations in *d* dimensions is:
NP-hard in general Euclidean space *d* even for 2 clusters
NP-hard for a general number of clusters *k* even in the plane
If *k* and *d* (the dimension) are fixed, the problem can be exactly solved in time $O(n^{dk+1} \log n)$,
where *n* is the number of entities to be clustered

Conclusion: K-means algorithm is useful for undirected knowledge discovery and is relatively simple. K-means has found wide spread usage in lot of fields, ranging from unsupervised learning of neural network, Pattern recognitions, Classification analysis, Artificial intelligence, image processing, machine vision, and many others.

Assignment No 6

Aim: Write a mobile app using Scala/ Python/ C++/ Android using Eclipse to beep the mobile speaker for three incorrect attempts of the password

Tools:
Eclipse, Android SDK, Java.

Theory:

This program deal with the giving simple beep when user attempts for three wrong inputs to the system.

Starting a New Project

Eclipse is a powerful, open source, integrated development environment (IDE) that facilitates the creation of desktop, mobile, and web applications. Eclipse is a highly versatile and adaptable tool. Many types of applications and programming languages can be used by adding different "plug-ins." For example, plug-ins are available for a very large number of programming languages as diverse as COBOL, PHP, Java, Ruby, and C++, to name a few. Additionally, plug-ins provide the capability to develop for different platforms, such as Android, Blackberry, and Windows. Many of the tools in the Eclipse IDE will be explained through the act of developing an Android app.

Android is a mobile operating system designed for smartphones and tablets. The operating system is very powerful, enabling access to a diverse set of hardware resources on a smartphone or tablet. Android is provided by Google and is continually updated, improved, and extended. This makes the development of apps for Android smartphones and tablets both exciting and challenging. As with Eclipse, the many features of the Android environment are best explained through the act of developing an app.

Setting Up the Workspace

Eclipse uses the concept of a workspace for organizing projects. Because Eclipse can be used to develop many types of applications, this is very useful. A workspace, in reality, is just a folder on some drive on your computer. The folder contains the application's code and resources, code libraries used by the application (or references to them), and metadata that is used to keep track of environment information for the workspace.

To begin, run Eclipse. The Workspace Launcher dialog window opens, asking which workspace you want to use. The default workspace (or last used) is displayed in the dialog window's text box. Most

IDEs are designed with the idea that developers are going to be working on the same machine each time they work on a project. This can cause problems in the education environment where students do not have the ability to work on the same machine and/or store their work on the machine they are currently working on. If you are using your own machine, you can skip to the next section; your workspace was created when you installed Eclipse and is ready to go. However, if you are working in an environment where you cannot use the same machine each time, you need to set up a workspace on either a flash drive or on a network drive. Determine which of these options is best for your situation and perform the following steps:

1. Create a folder in your selected location named **workspace**.
2. Go back to the Workspace Launcher and browse to your new folder. Click OK.

   Often in a situation where you change the workspace to a location not on the machine that Eclipse is installed on, Eclipse will not be able to find the Android SDK. If it cannot find the SDK, a dialog window opens. If this happens, you will have to tell Eclipse where the files are located by performing the next steps.

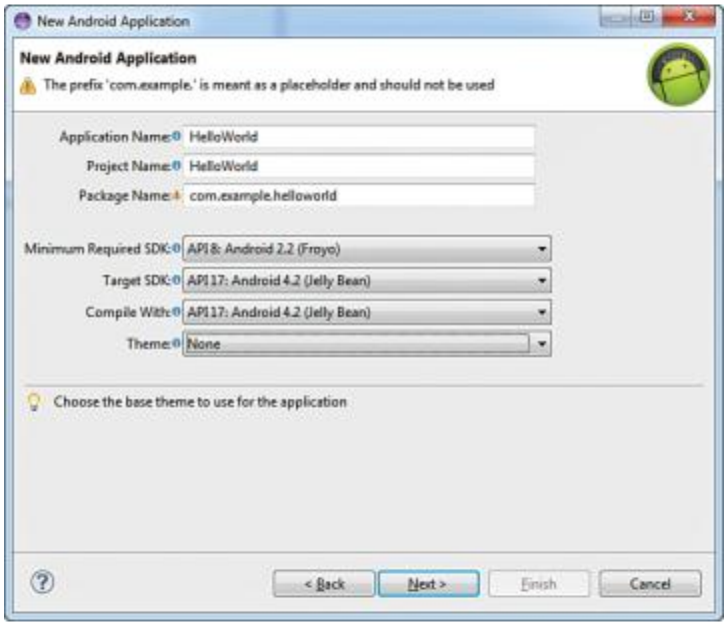
3. Click Open Preferences on the dialog window and browse to the sdk folder. This is usually located in the .android folder. Click Apply.

   The available Android versions should be displayed in the window.

4. Click OK to close the dialog window. Your workspace is now ready to begin Android development.

Creating the Project

The traditional beginning tutorial for many different languages and development platforms is "Hello World." Your first Android app will be a slightly modified "Hello World" app. In Eclipse, all Android apps are created within a project. To create your first app, you will have to create your first project. Creating a new project requires stepping through a series of windows and making choices to configure your app. To get started, from Eclipse's main menu choose File > New > Android Application Project. You should see the New Android Application dialog window, as shown in Figure 3.1.


New Android Application

New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: HelloWorld
Project Name: HelloWorld
Package Name: com.example.helloworld

Minimum Required SDK: API 8: Android 2.2 (Froyo)
Target SDK: API 17: Android 4.2 (Jelly Bean)
Compile With: API 17: Android 4.2 (Jelly Bean)
Theme: None

Choose the base theme to use for the application

Initial new Android application window configured for “Hello World.”

Fill out the screen as shown. The application name is displayed on the phone’s screen as the name of the app. You can use spaces if you want. As you type the name, the project name and package name will be completed. There are no spaces allowed in these items. The wizard will remove them as you type. Don’t put them back in either of these fields. The package name is important. For this initial project you don’t need to change the default. However, if you are building an app for sale, in place of “example” you should put your company name. This identifier will be used in the Play Store to link your apps to the services they use and connect all your apps.

Next, click the Minimum Required SDK drop-down. A list of potential Android SDKs are listed. SDK stands for Software Development Kit, and it is a set of tools and code libraries used to write software for a specific platform. Each release of the Android OS is associated with an SDK so that programmers can write code for that platform. An application programming interface (API) is a set of routines that allow a program (app) to access the resources of the operating system to provide functionality to the user. The minimum required SDK determines what phones and other Android devices will be able to install your app. (Phones and tablets using Android operating systems earlier than this selection will not even see your app in the Play Store.) This selection will also determine the features you can program into your app. The recommended minimum is the default: *Froyo API 8*. An app that has this minimum will be accessible to more than 90% of the devices “in the wild.”

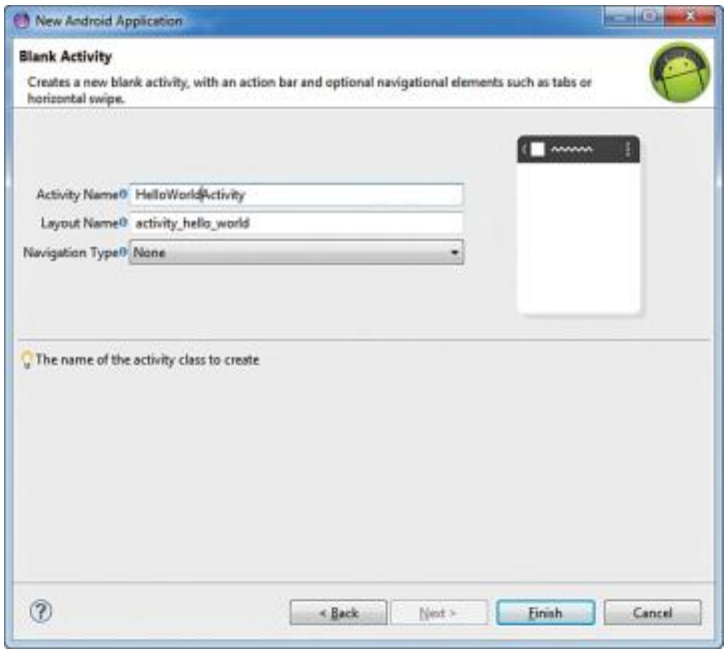
The Target SDK should usually be set to the latest version of the Android operating system. At the writing of this book, that version is the Jelly Bean (API 17). After you release an app, you should periodically update these values and recompile your app as new versions of Android are released. At times, new versions of the operating system can affect the performance of your app, so it is best to keep the app up to date. The Compile With target should also be the latest SDK.

Themes are a useful way to ensure a consistent look for your app. However, because this is an introduction you will not be using them in this book. Click the drop-down and select None as your theme.

The Configure Launcher Icon window allows you to associate an icon with your app that will be displayed on the phone's screen along with the app name. Notice the different sizes of the icons. If you are providing an icon for your app, you will have to supply several sizes of the same picture. This is because Android apps can run on any Android device that meets the app's SDK requirements. However, these devices can have different screen resolutions and different screen sizes. By supplying different icon sizes, the app will pick the one that best matches the device it is running on. This helps ensure that your app will show up as you design it, regardless of the characteristics of the device it is running on. Suggested sizes for app icons are 32×32, 48×48, 72×72, 96×96, and 144×144 pixels for low to extra high density screens. Accept the default icon for this app by clicking the Next button.

The Create Activity window is the next step in configuring your project. An Activity is a core component of any Android application. Activities are typically associated with a visible screen. Most of the core functionality of an app is provided by an activity and its associated screen (called a *layout*). Click among the different activity options. Notice that when you have selected some of them, the Next button is disabled. The choices are limited by your choice of minimum and target SDK. Eclipse won't let you use features that will not work on the devices you targeted. In this case, because you selected API 8 as the minimum SDK that your app would be allowed to run on, some activity types are not available, even though they are available in the target SDK you selected.

From the list of possible activities, choose Blank Activity and click the Next button. The Blank Activity window is displayed (Figure 3.2). This allows us to configure the first Activity in our app. With this screen we can change the name of the activities we create. In the Activity Name text box, delete MainActivity and type HelloWorldActivity. Notice below Activity Name is Layout Name. As you typed in the activity name, the text in this box changed to reflect the text you entered. A layout is an XML file that provides the user interface for the activity. Layouts are discussed in detail later. For now, just remember that every activity has an associated layout file.



Blank Activity

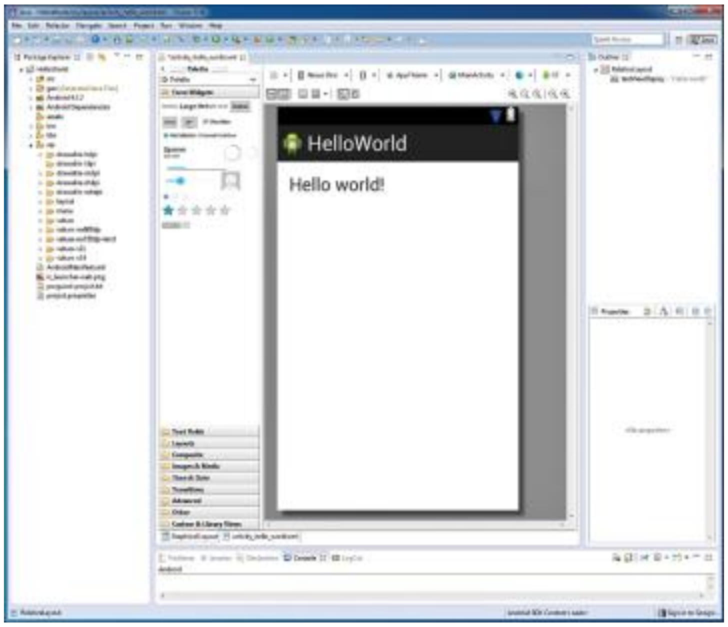
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.

Activity Name: HelloWorldActivity
Layout Name: activity_hello_world
Navigation Type: None

The name of the activity class to create

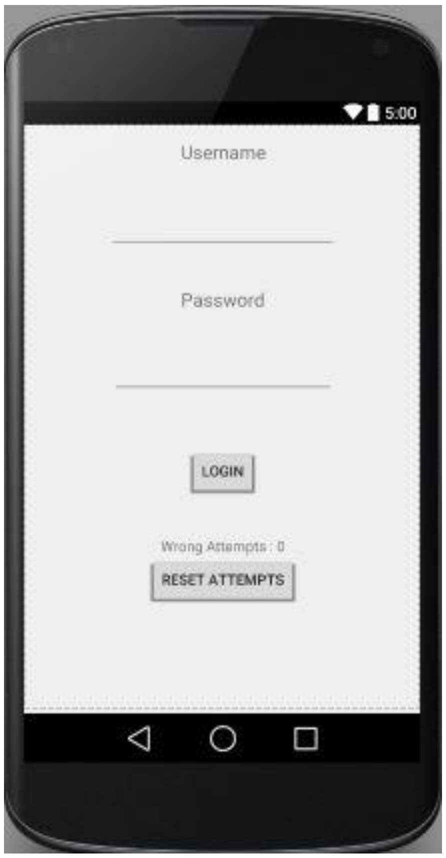
Blank Activity window with default selections.

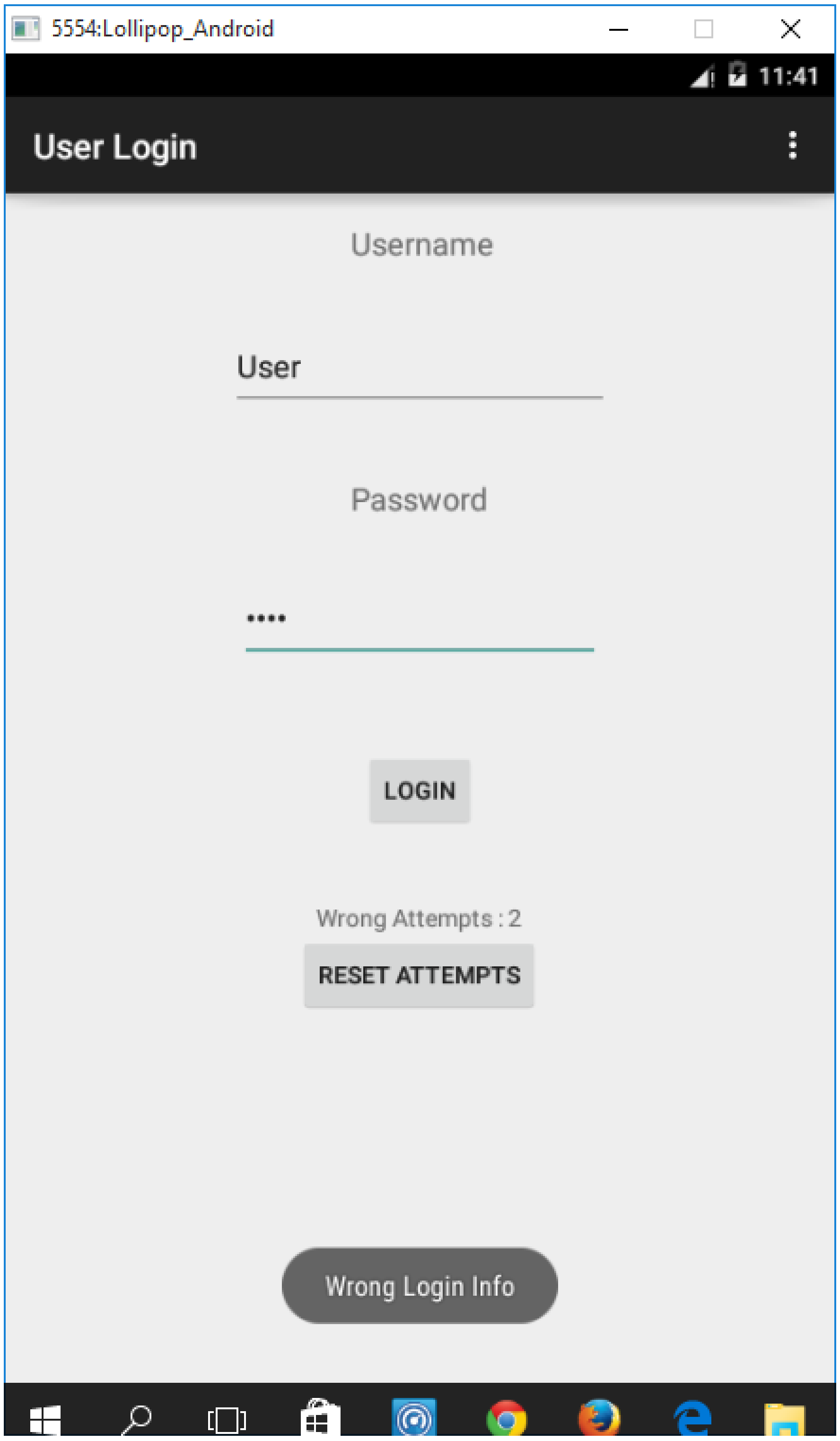
The final item on this page is Navigation Type. Select it and click among the options. Notice that just like the Create Activity window, you are not allowed to use some navigation types. Again this is based on the SDK choices you made earlier. Select None as your Navigation Type and click Finish. Your app project is created! Depending on the capability of your computer, it may take some time to create the project. When Eclipse has finished creating your project, your Eclipse environment should look like

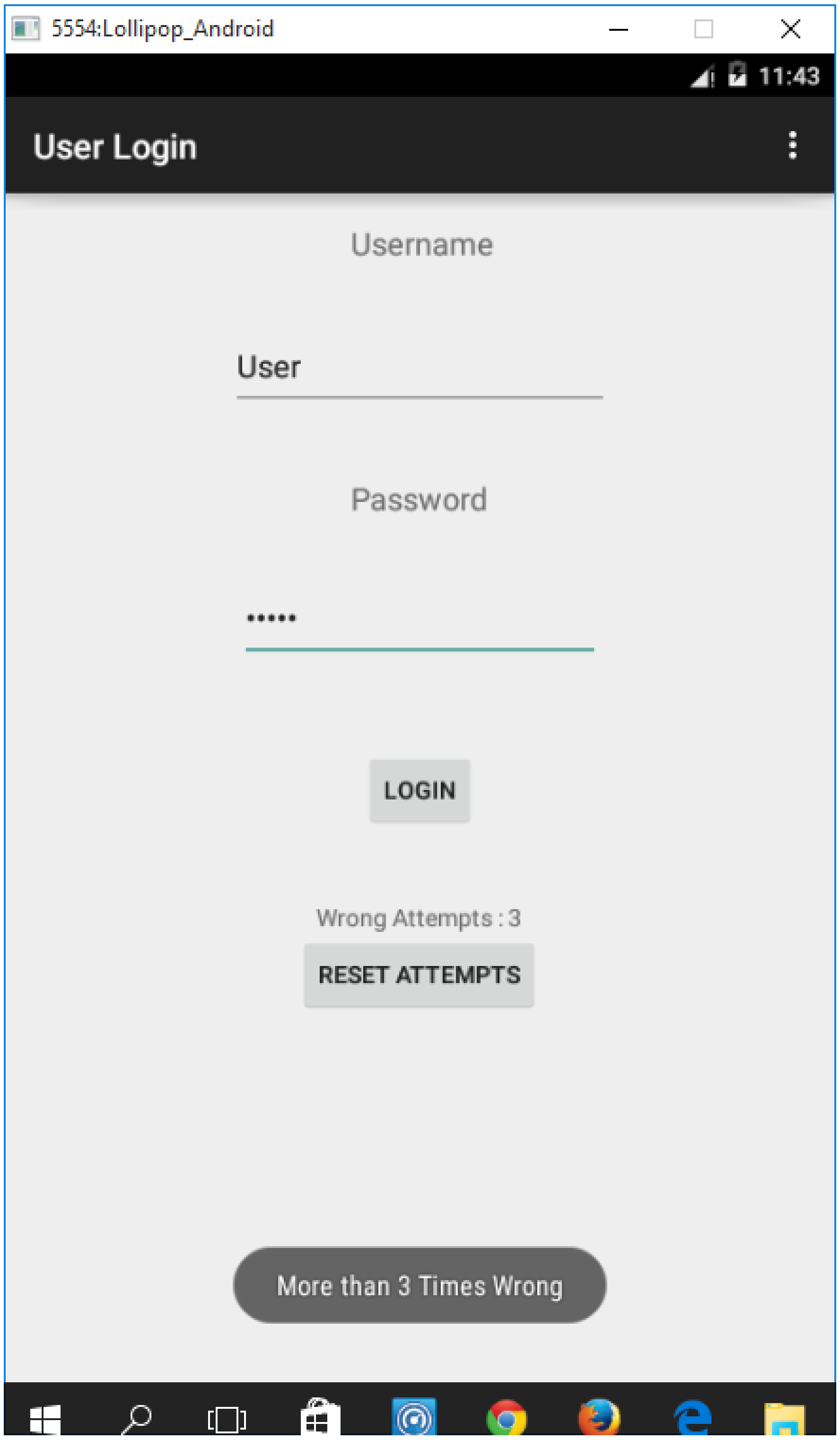


Eclipse with the newly created Hello World project.

Screenshot:

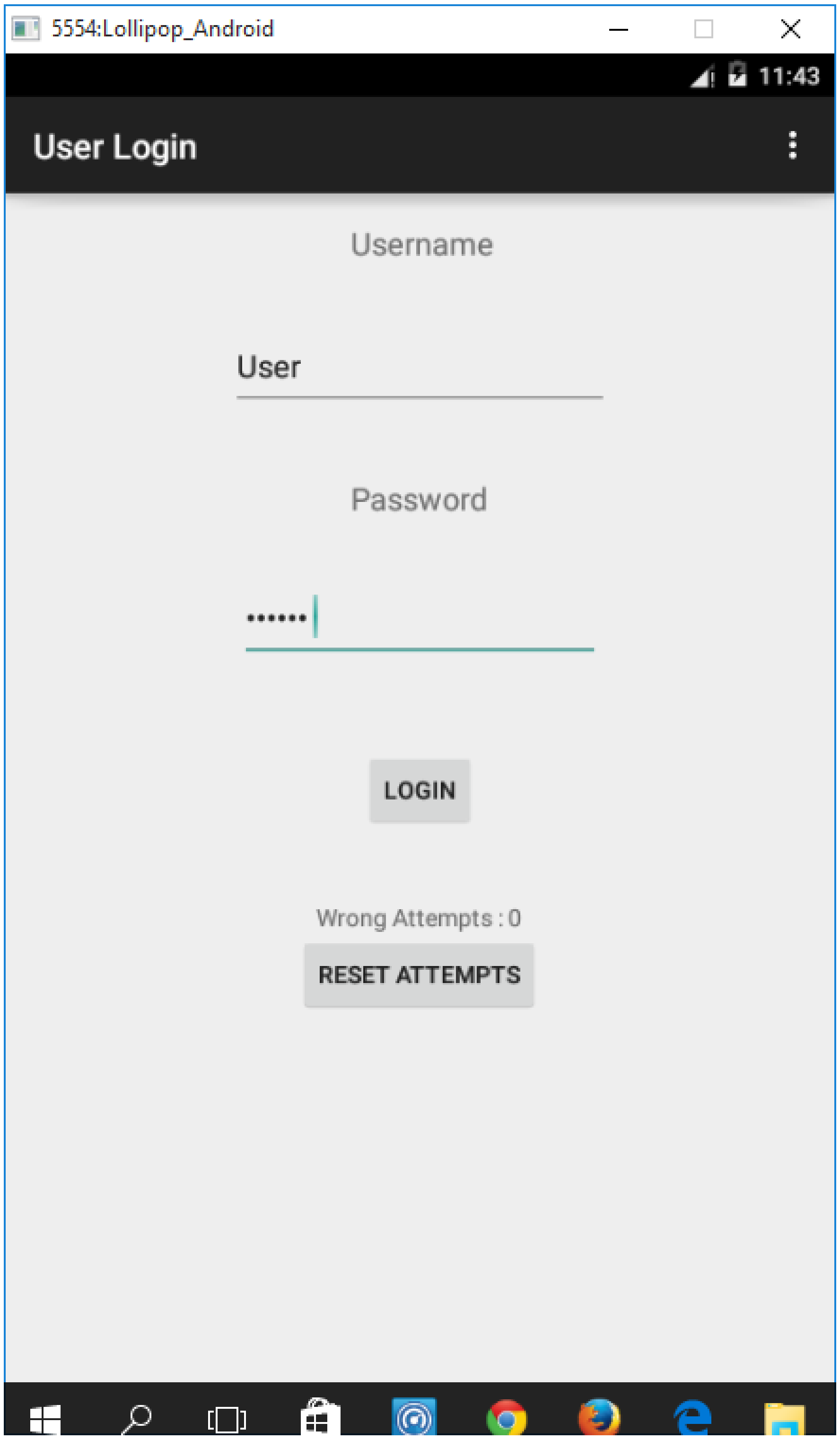

5:00
Username
Password
LOGIN
Wrong Attempts : 0
RESET ATTEMPTS

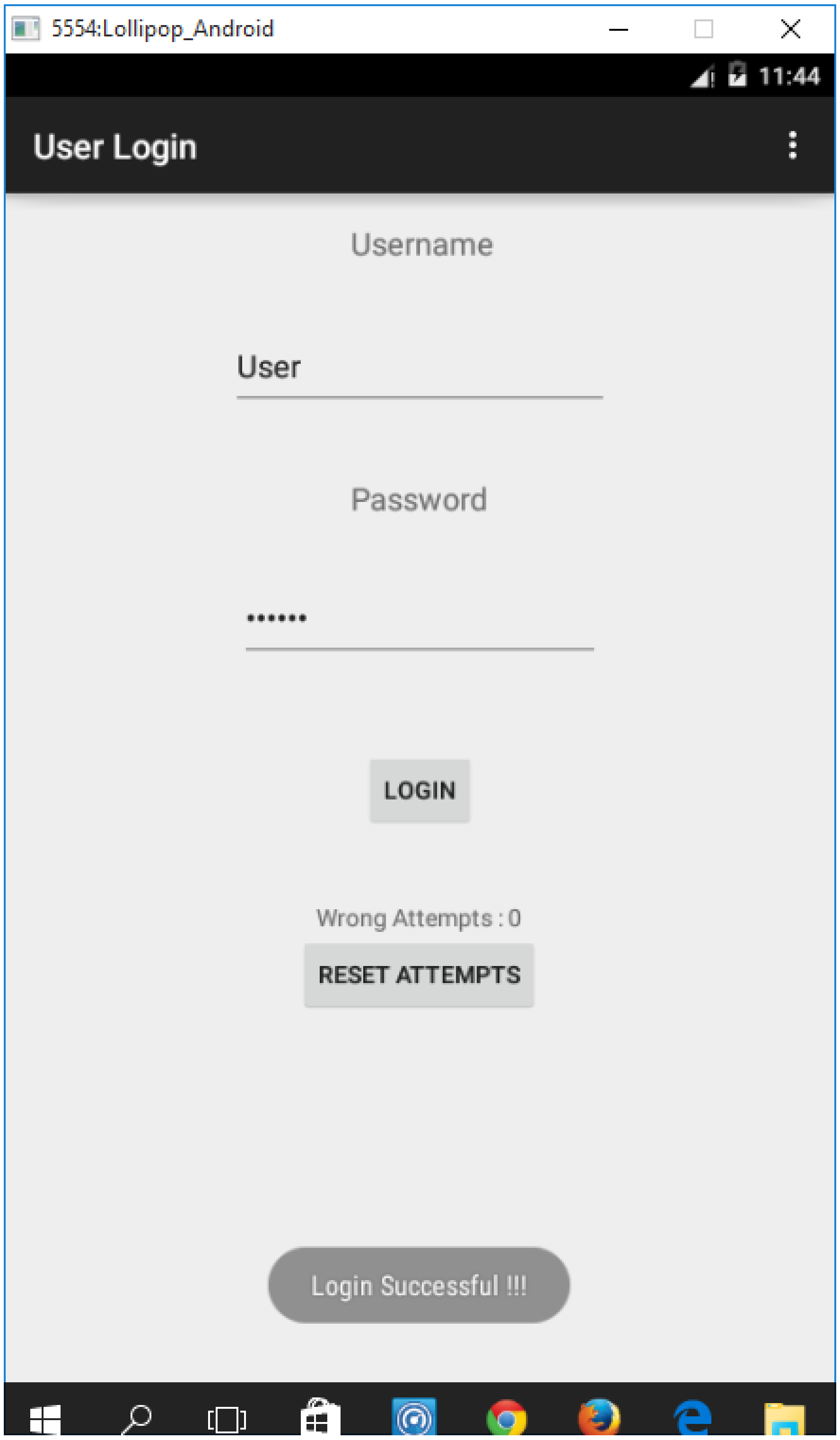

5554:Lollipop_Android
11:41
User Login
Username
User
Password
••••
LOGIN
Wrong Attempts : 2
RESET ATTEMPTS
Wrong Login Info

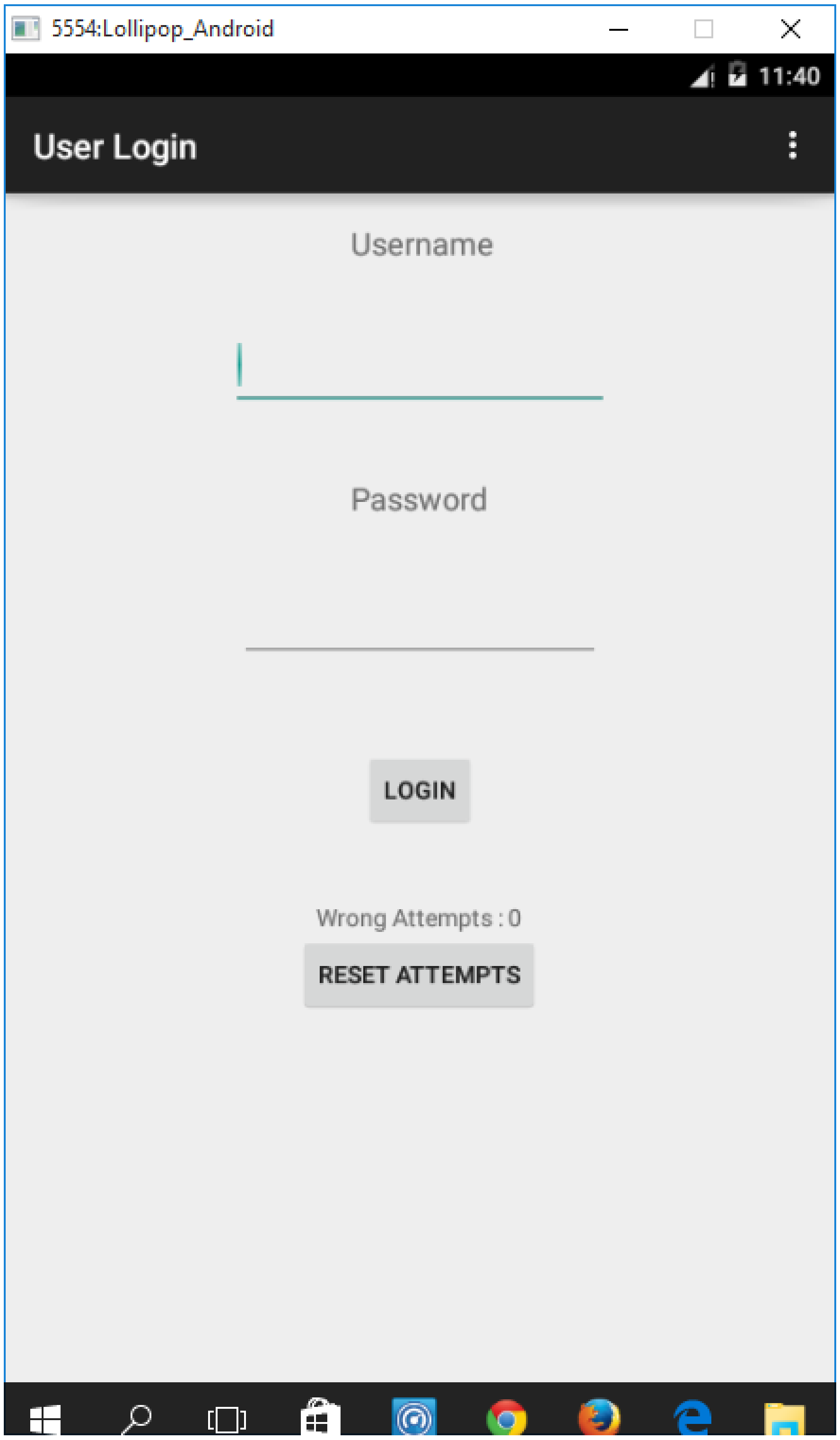


Wrong Attempts : 3

More than 3 Times Wrong


5554:Lollipop_Android
11:43
User Login
Username
User
Password
••••••
LOGIN
Wrong Attempts : 0
RESET ATTEMPTS


5554:Lollipop_Android
11:44
User Login
Username
User
Password
••••••
LOGIN
Wrong Attempts : 0
RESET ATTEMPTS
Login Successful !!!


5554:Lollipop_Android
11:40
User Login
Username
Password
LOGIN
Wrong Attempts : 0
RESET ATTEMPTS

Program:

```
package com.example.dipu.userlogin;

import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


public class MainActivity extends ActionBarActivity {

    Button Login, Reset;
    TextView Attempt;
    int i=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Login = (Button) findViewById(R.id.button);
        Attempt = (TextView)findViewById(R.id.textView3);
        Login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                i++;
                EditText U = (EditText) findViewById(R.id.editText);
                EditText P = (EditText) findViewById(R.id.editText2);
                String User = U.getText().toString();
                String Pass = P.getText().toString();

                if (User.equals("User") && Pass.equals("Secret"))
                {
                    Toast.makeText(getApplicationContext(), "Login Successful !!!",
Toast.LENGTH_SHORT).show();
                    Reset();
                }
                else {
                    if (i < 3)
                        Toast.makeText(getApplicationContext(), "Wrong Login Info",
Toast.LENGTH_SHORT).show();
                    else
                    {
                        Toast.makeText(getApplicationContext(), "More than 3 Times Wrong",
Toast.LENGTH_SHORT).show();
                        Uri notification =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
                        Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), notification);
                        r.play();
                    }

                    String A = Attempt.getText().toString();
                    int k = 17, Ano = 0;

                    for (; k < A.length(); k++)
                        Ano = Ano * 10 + (int) (A.charAt(k) - 48);
                    String UpdatedA = A.substring(0, 17) + ++Ano;
                    Attempt.setText(UpdatedA.toCharArray(), 0, UpdatedA.length());
                }
            }
        });

        Reset = (Button)findViewById(R.id.button2);
        Reset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Reset();
            }
        });
    }

    void Reset() {

        String A = "Wrong Attempts : 0";
        Attempt.setText(A.toCharArray(), 0, A.length());
        i=0;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Conclusions: In this way, we have studied the development of mobile app using Scala/ Python/ C++/ Android using Eclipse to beep the mobile speaker for three incorrect attempts of the password

Group C

Assignment No 1

Aim:- Write a program to Smart Watch App Development with Tizen

Objectives:- . Objective of this assignment is to design simple comic app with the Tizen SDK for Wearable and run it on the smart watch emulator that comes bundled with the IDE.

Tools Requirement:-
64-bit windows 7 OS with 64-bit Intel-i5/i7,
tizen-sdk_2.3.63_windows-64,
tizen-wearable-sdk-2.2.159_windows-64,
tizen-wearable-sdk-image-TizenSDKW_1.0.0-windows64,
HTML program.

Hardware: - 64-bit window 7

Software: -
tizen-sdk_2.3.63_windows-64,
tizen-wearable-sdk-2.2.159_windows-64,
tizen-wearable-sdk-image-TizenSDKW_1.0.0-windows64,
HTML program.

Theory:-
Definition:

Tizen is an operating system of the Linux family, targeting a range of devices from smartphones to smart watches and a lot more. While Tizen is a project within the Linux Foundation, it is guided by the Tizen Association, whose members include Samsung, Intel, and other well known companies in the technology industry.

How to install and configure the Tizen SDK for Wearable and develop a smart watch application with the IDE. Let's get started.

Installing & Configuring the SDK

Step 1: Tizen SDK or Tizen SDK for Wearable?

Currently, two types of SDKs are available, Tizen SDK and Tizen SDK for Wearable. Since this tutorial is about developing a standalone smart watch app, what you need is Tizen SDK for Wearable.

You can download it from the Tizen Developers website. You need to download an appropriate **install manager** that matches your operating system and version. If you prefer an offline installation to an online one, you need to download an **SDK image** too. If your operating system is Windows 8 or Windows 8.1, you can download the installation files categorized under Windows 7. They will work on Windows 8 and 8.1 just fine.

Step 2: Requirements

Refer to Tizen's detailed instructions to read about the hardware and software requirements your computer should meet.

You can install the SDK even if your computer doesn't meet these hardware requirements. However, if you do, the smart watch emulator will be slow, resulting in poor app testing. Visit the documentation for more details. It explains how to enable **Virtualization Technology** (VT) in your BIOS and **Data Execution Prevention** on Windows.

Step 3: Installing the SDK

1. Run the **install manager** you downloaded earlier. This is an .exe file with a filename like tizen-wearable-sdk-2.2.159_windows64.exe, depending on your operating system and version.
2. Click **Advanced** to go to the next screen.
3. In that screen, check the **SDK image** radio button and navigate to the zip file containing the appropriate SDK Image. Note that I'm assuming that you prefer an offline installation and you have already downloaded the necessary SDK image to your development machine.
4. Select the **SDK image** zip file and click **Open** in the dialog box.
5. An **Extracting SDK Image** message will appear. Click **OK** after finishing the extraction.
6. Click **Next** and the **License Agreement** window will appear.
7. Agree to the license and click **Next.**
8. The **configuration window** will appear next. I recommend leaving all the check boxes checked and clicking **Next**.
9. Finally, when the **install manager** asks for a location for the installation, specify your choice by selecting a path and clicking **Install.**
10. If you have already configured your BIOS correctly, **Intel Hardware Accelerated Execution Manager** (Intel HAXM) will also be automatically installed during this process. If not, quit the installation process and configure the BIOS so that it can allow Intel HAXM to be installed
11. Don't forget to enable **Data Execution Prevention** if your operating system is Windows. Resume the installation.

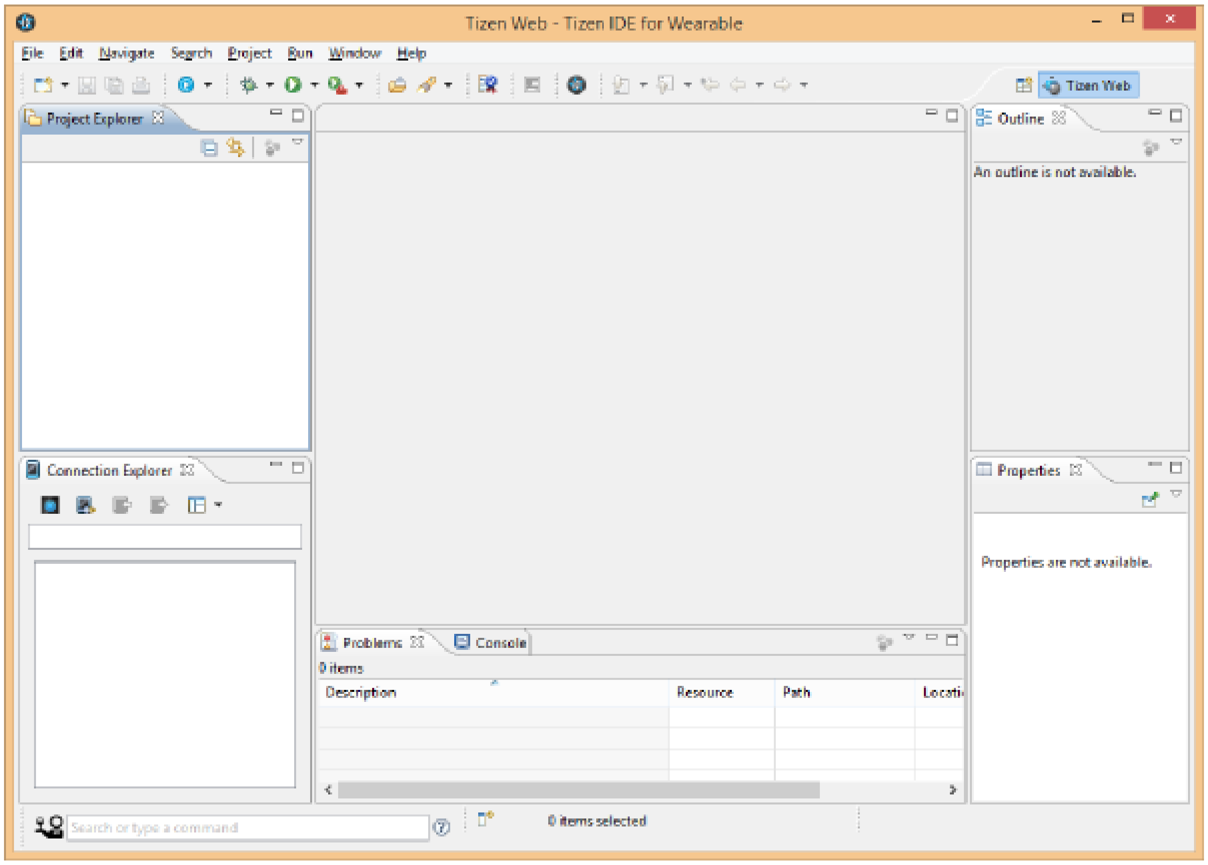
you can install **Intel HAXM** separately. Once the installation is finished, restart your computer.

Step 4: Configuring the IDE

1. Browse to the folder in which you have installed the SDK and navigate to the **ide** subfolder. Run the executable file named **IDE**.
2. After a few minutes, a window will appear, asking for a location for the **workspace** to save the apps you develop. Specify a path of your choice for the location and click **OK**. After the configuration, the IDE should appear.

Step 5: Features of the IDE

On the left pane of the IDE, there are two windows, **Project Explorer** and **Connection Explorer**. The **Project Explorer** shows the projects created by the user. The **Connection Explorer** lists the connected devices that are currently available, emulator instances or remote devices.



An outline is not available.

Properties are not available.

0 items

0 items selected

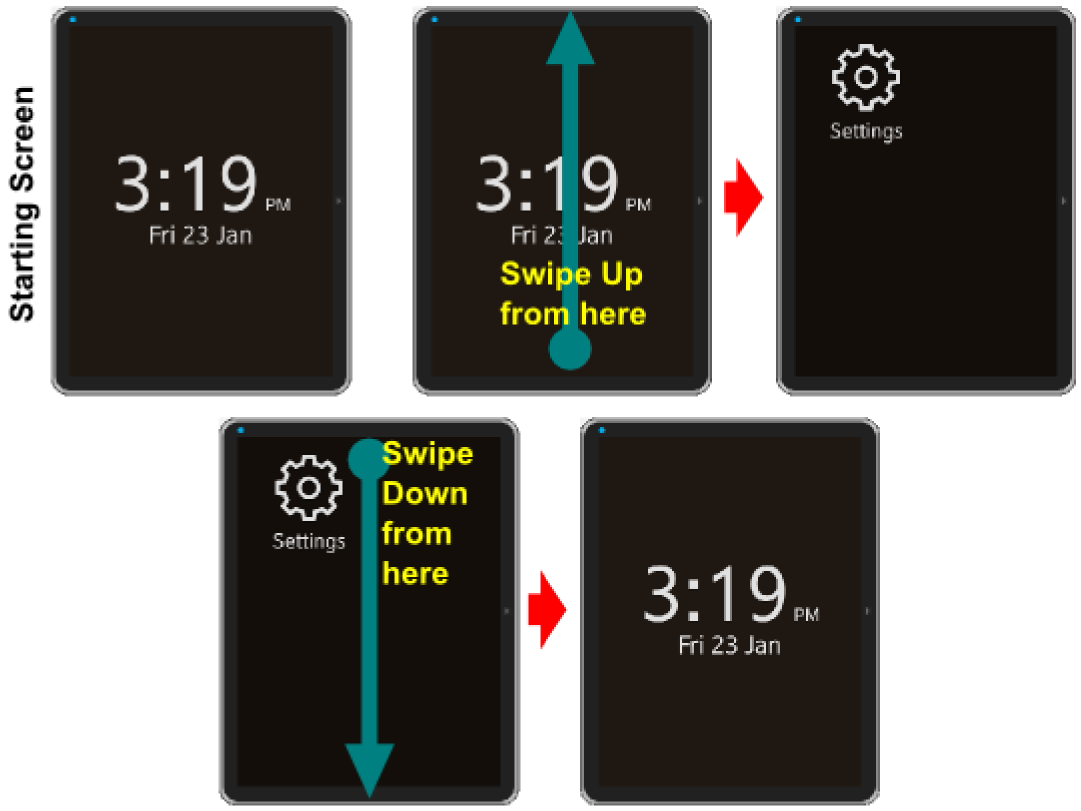
Step 6: Creating an Emulator Instance

1. In the **Connection Explorer,** click on the **Emulator Manager** icon, the leftmost blue button.
2. Click **Yes** in the **User Account Control** window that appears. This will bring up the **Emulator Manager** window.
3. Click **Add New** and give the emulator instance a name.
4. When you click **Confirm,** the new emulator instance will be created. Click the blue button with an arrow in the emulator icon to launch the emulator.

It will take some time to launch the emulator. You should see a window with a starting screen similar to the below screenshot when it is up and running. The emulator instance should appear as an entry in the **Connection Explorer**.

Swipe up from the bottom middle point of the starting screen to go to the screen that displays the installed apps on the device or emulator. Since you haven't installed any apps yet, only the **Settings** icon is displayed.

You can go back to the previous screen or quit an app by swiping down from the top middle of the screen.



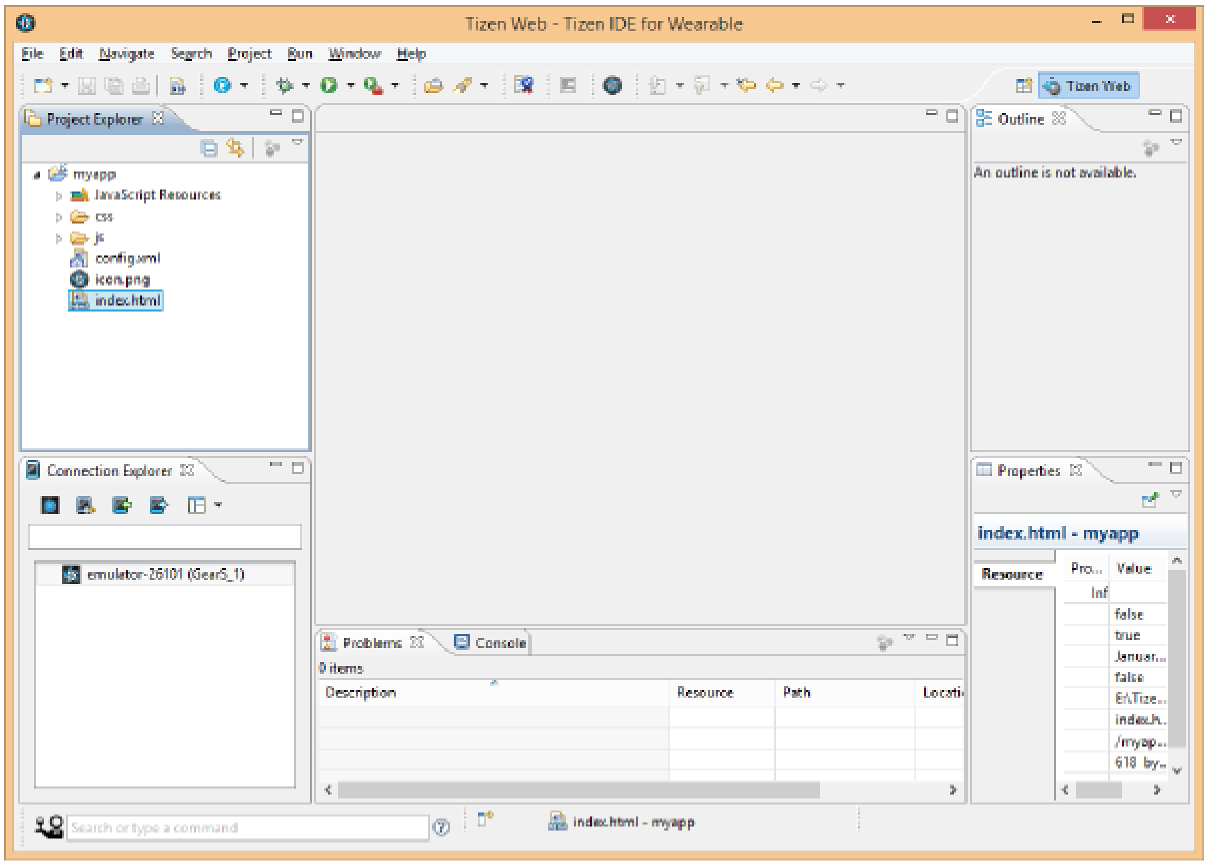
2. Developing a Simple Comic App

In this example, we're going to create a simple app to display a comic strip. Let's look at each step in turn.

Step 1: Creating a New Project

Let's create a new project in the IDE.

1. Go to **File** > **New** > **Tizen Wearable Web Project**.
2. In the window that appears, select **Basic** > **Basic application** and set the **Project name** to **myapp**.
3. Tick the **Use default location** check box or browse to a different location of your choice, and click **Finish**.
4. Your new project, **myapp**, should appear in the **Project Explorer**.
5. Click the small arrow on the left of **myapp** to expand the project structure.
6. You should see an **index.html** file, a **css** subfolder, a **js** subfolder, and a few other files and folders.



HTML, CSS, and JavaScript form the basis for programming on the Tizen platform. If you're an HTML wizard, then you don't have to learn a new programming language to write applications for the Tizen platform. You can use your existing HTML, CSS, and JavaScript skills.

Step 2: Adding Files, Assets, and Resources

We first need to add two subfolders to the **myapp** project, **comic** and **images**. To do this, right-click the **myapp** project folder in the IDE and select **New** > **Folder**. The subfolders should appear in the expanded **myapp** folder in the IDE.

Download the source files for this project from GitHub and navigate to the **images** subfolder, which contains a number of png files. Copy the png files to the **images** subfolder you created a moment ago.

You can paste files to the **images** subfolder in the **Project Explorer** window by right-clicking the subfolder and selecting **Paste** from the popup menu.

Next, create nine HTML files with the following file names in the **comic** subfolder by right-clicking the **comic** subfolder and selecting **New** > **File**. Make sure to include the **.html** extension for the files.

- **page1.html**
- **page2.html**
- **...**
- **page9.html**

You should now have nine HTML files in the **comic** subfolder.

Step 3: Adding Code
Let's now edit the code in **index.html**. This file is the entry point of your application. Double-click **index.html** to open the file in the IDE. Change the contents of the <title> tag to <title>2nd Race</title>. Next, change the contents of the <body> tag with the following:

All we did, is add an image to the page and two buttons to navigate to the other pages since our comic will have ten pages. Once you have made these changes, save the file by selecting **File > Save** from the menu.

If you are new to HTML and CSS, Tuts+ has a huge collection of excellent tutorials that will help you get up to speed with the basics of web development.

Next, double-click **style.css** in the **css** subfolder and change its contents as shown below.

We've added some styling for body, images, and the navigation menus. Once you have made the changes, save the file.

Similarly, add code to all the other HTML files you have created. The **style.css** file in the **css** subfolder must be externally linked to all of these HTML files. If you're note sure about this step, then download the source files from GitHub and examine the source for clarification.

Step 4: Testing the App

To test your app, select the **myapp** project folder and, from the menu, choose **Project > Build Project** to build the project. Make sure the emulator instance is up and running.

Right-click the **myapp** folder and select **Run As > Tizen Wearable Web Application** to run the project in the emulator. Use the arrow buttons in the user interface to navigate to the next or previous page. Swipe down from the top of the screen to quit the app.

Program with proper indentation & comments & output :

Comic Html Pages:

PAGE 1:
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>

<body>
  <div>
            <div><img src="../images/page1.png" alt="Page 1" /></div>
            <div><a href="../index.html" class="btn" >&lt;&lt;</a>
                        <a href="page2.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 2:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>

<body>
  <div>
            <div>  <img src="../images/page2.png" alt="Page 2" /></div>
            <div><a href="page1.html" class="btn" >&lt;&lt;</a>
            <a href="page3.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 3:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
```

```
</head>

<body>
  <div>
            <div>  <img src="../images/page3.png" alt="Page 3" /></div>
            <div><a href="page2.html" class="btn" >&lt;&lt;</a>
             <a href="page4.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 4:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>
<body>
  <div>
            <div>  <img src="../images/page4.png" alt="Page 4" /></div>
            <div><a href="page3.html" class="btn" >&lt;&lt;</a>
             <a href="page5.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 5:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>
```

```
<body>
  <div>
           <div>  <img src="../images/page5.png" alt="Page 5" /></div>
           <div><a href="page4.html" class="btn" >&lt;&lt;</a>
            <a href="page6.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 6:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>
<body>
  <div>
           <div>  <img src="../images/page6.png" alt="Page 6" /></div>
           <div><a href="page5.html" class="btn" >&lt;&lt;</a>
            <a href="page7.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

PAGE 7:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>2nd Race</title>
    <link rel="stylesheet" type="text/css" href="../css/style.css"/>
    <script src="../js/main.js"></script>
</head>
<body>
  <div>
           <div>  <img src="../images/page7.png" alt="Page 7" /></div>
```

```
            <div><a href="page6.html" class="btn" >&lt;&lt;</a>
            <a href="page1.html" class="btn" >&gt;&gt;</a> </div>
  </div>
</body>
</html>
```

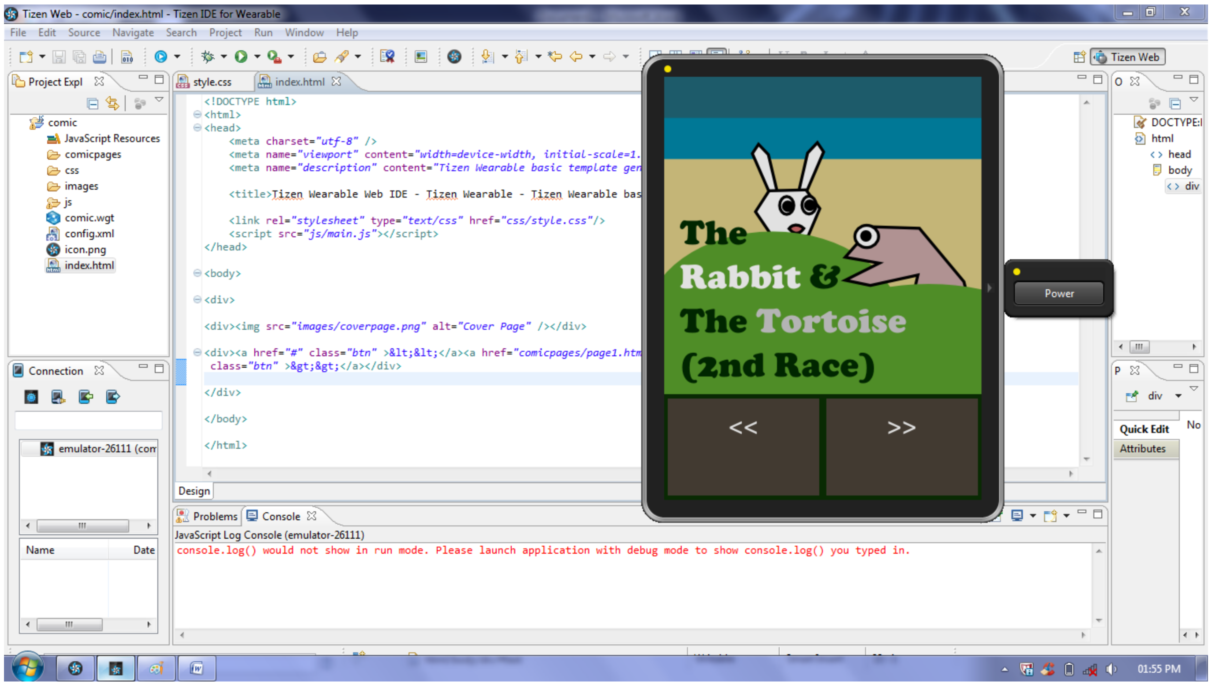
Index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">
    <meta name="description" content="Tizen Wearable basic template generated by Tizen Wearable
Web IDE"/>

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable basic Application</title>

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>
<div>
  <div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.html"
 class="btn" >&gt;&gt;</a></div>

</div>
</body>
</html>
```

Style.css

```
* {
    font-family: Verdana, Lucida Sans, Arial, Helvetica, sans-serif;
}

body {
    margin: 0px auto;
    background-color:#0a3003;
}

img {
  margin: 0;
  padding: 0;
  border: 0;
  width: 100%;
  height: auto;
  display: block;
  float: left;
}


.btn {
  display: inline-block;
  padding: 15px 4% 15px 4%;
  margin-left: 1%;
  margin-right: 1%;
  margin-top: 5px;
  margin-bottom: 5px;
  font-size: 30px;
  text-align: center;
  vertical-align: middle;
  border: 0;
  color: #ffffff;
  background-color: #4b4237;
  width: 40%;
  height: 80px;
  text-decoration: none;}
```

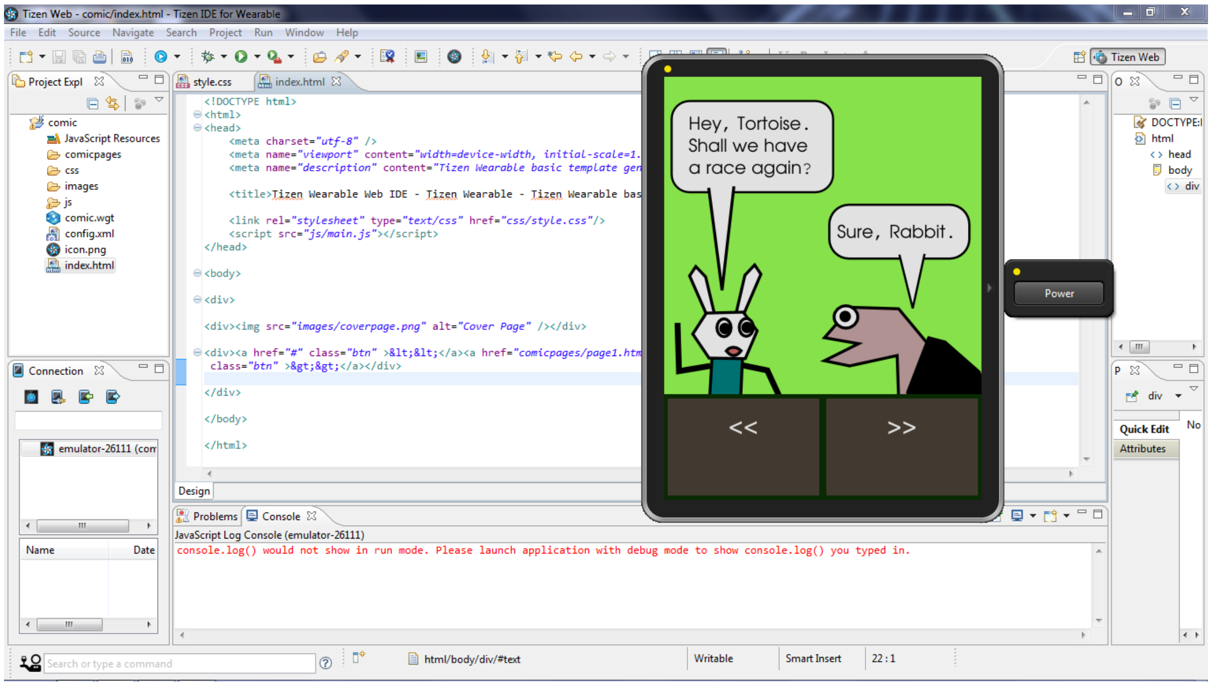
Case study (Any example):



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



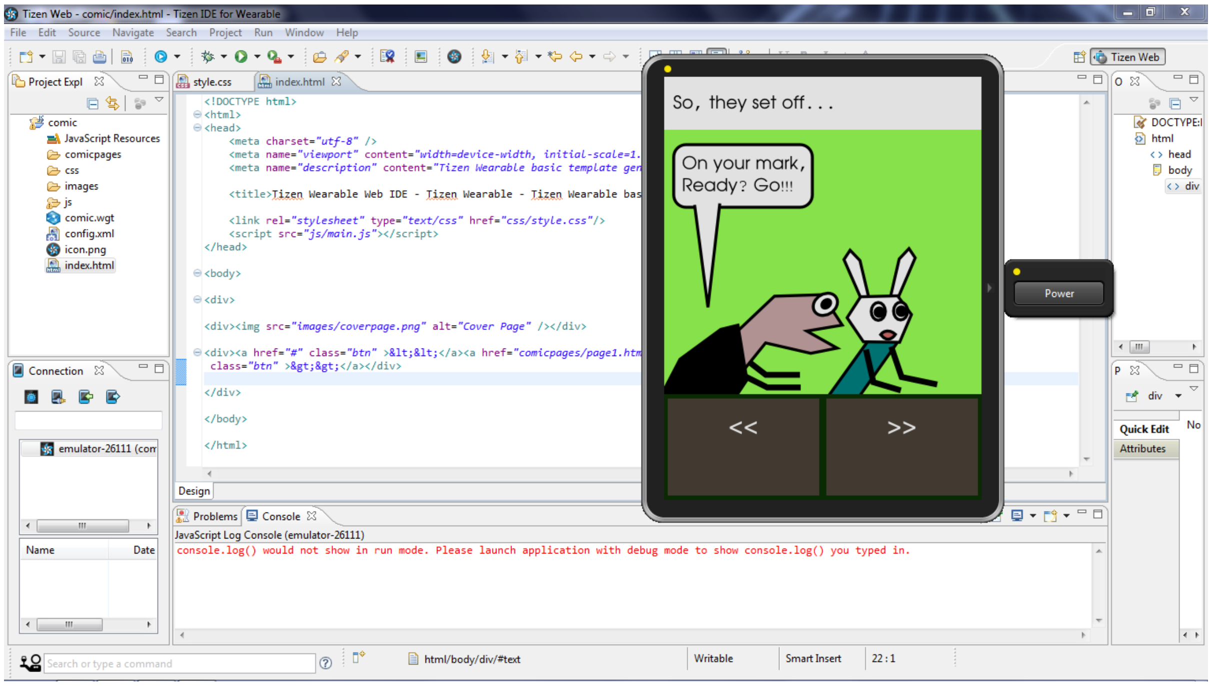
Tizen Web - comic/index.html - Tizen IDE for Wearable

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```

html/body/div/#text | Writable | Smart Insert | 22 : 1



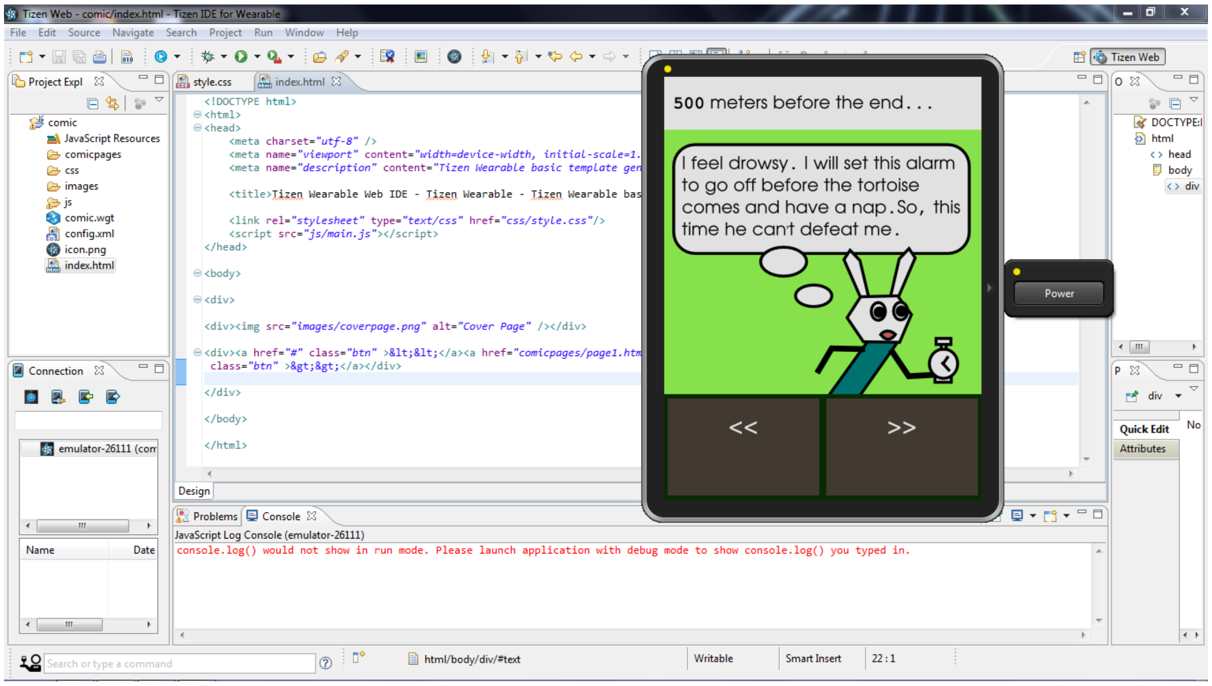
Tizen Web - comic/index.html - Tizen IDE for Wearable

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

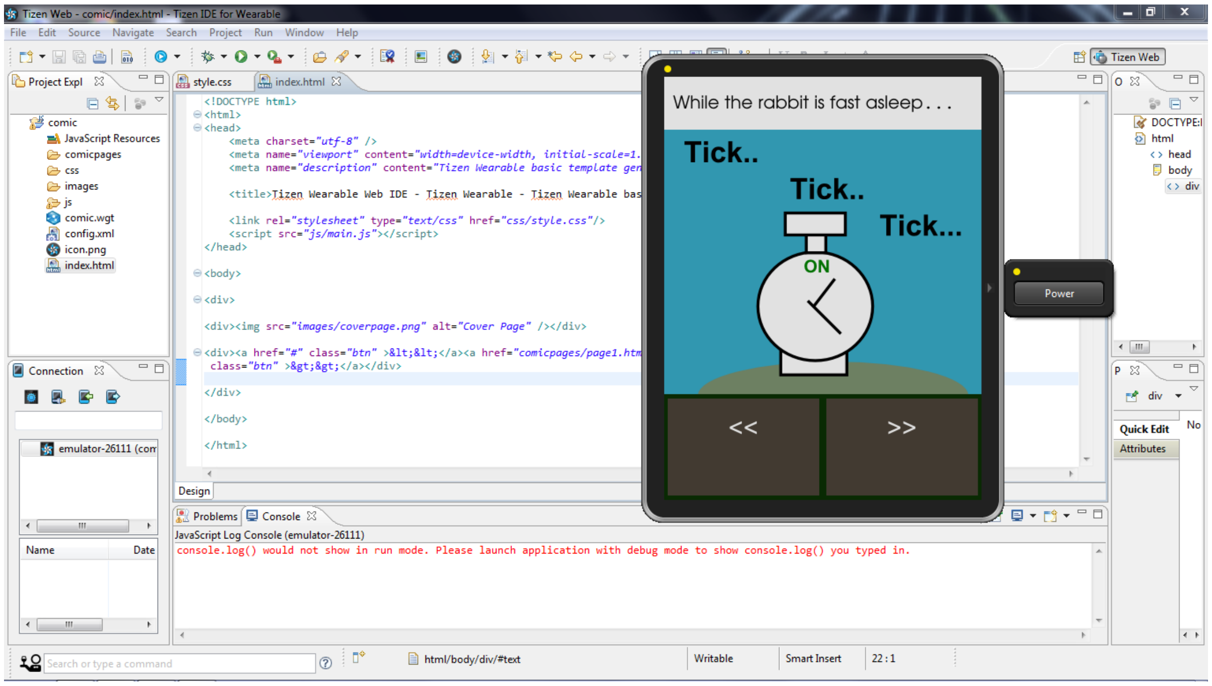
```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```

html/body/div/#text | Writable | Smart Insert | 22 : 1



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

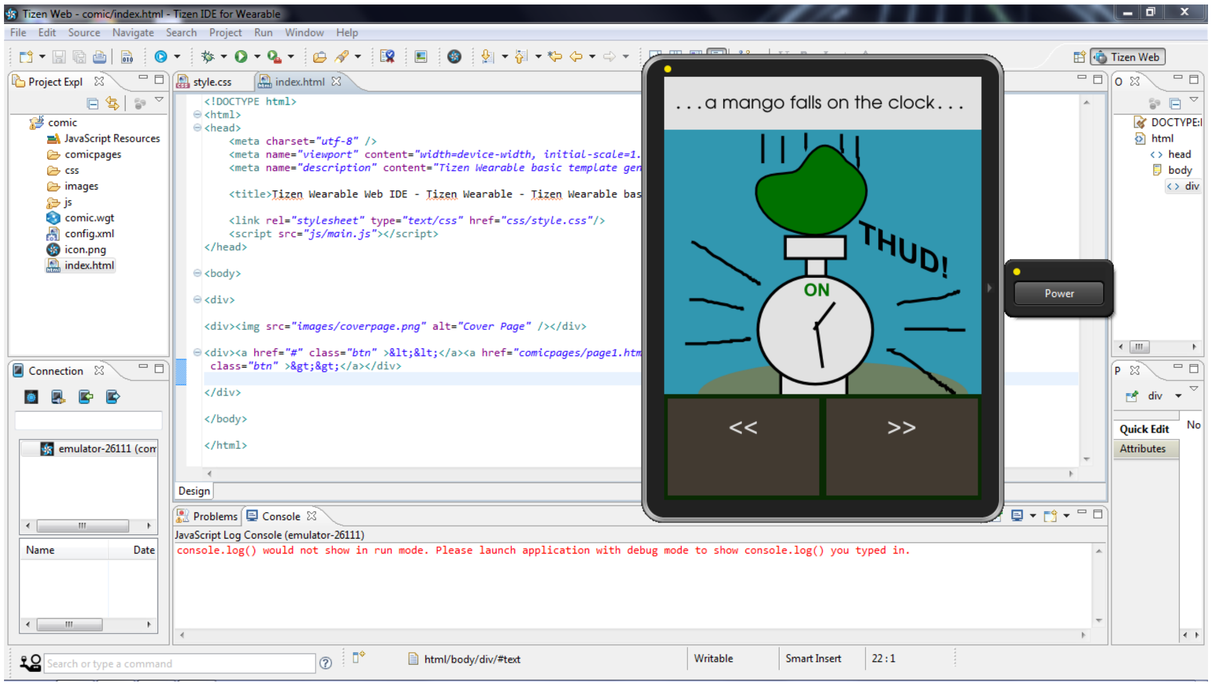
```
JavaScript Log Console (emulator-26111)
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



Tizen Web - comic/index.html - Tizen IDE for Wearable
While the rabbit is fast asleep . . .
Tick..
Tick..
Tick...
ON
<<
>>
Power

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

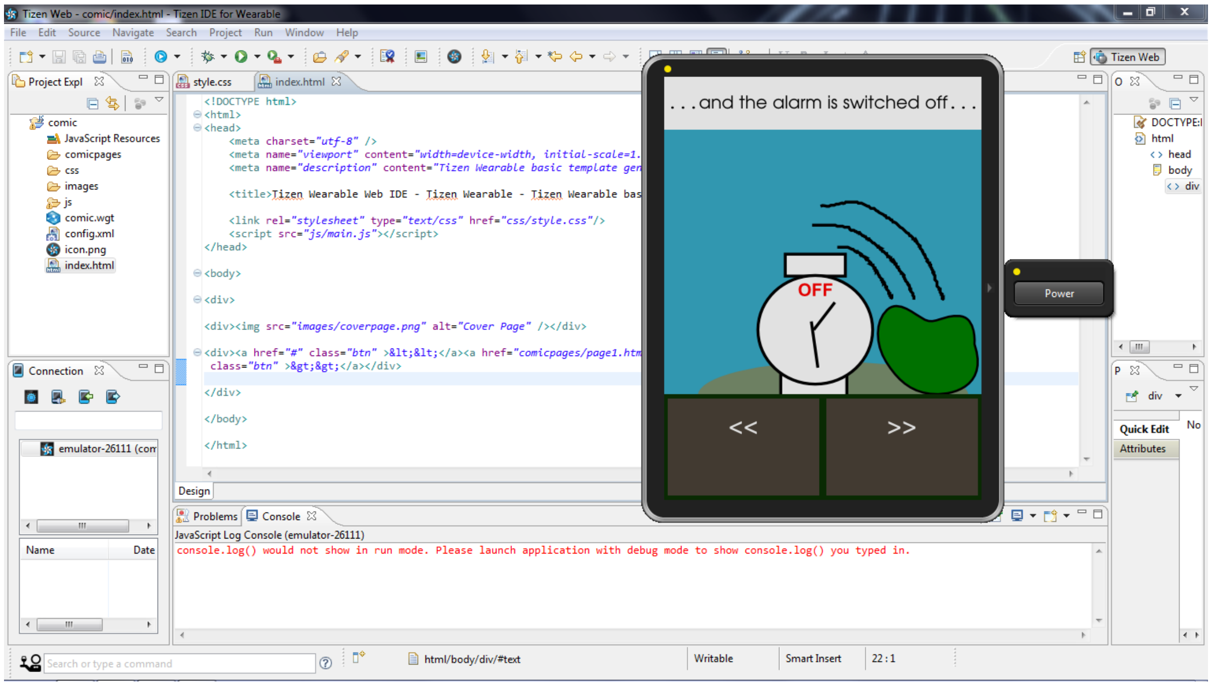
```
JavaScript Log Console (emulator-26111)
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

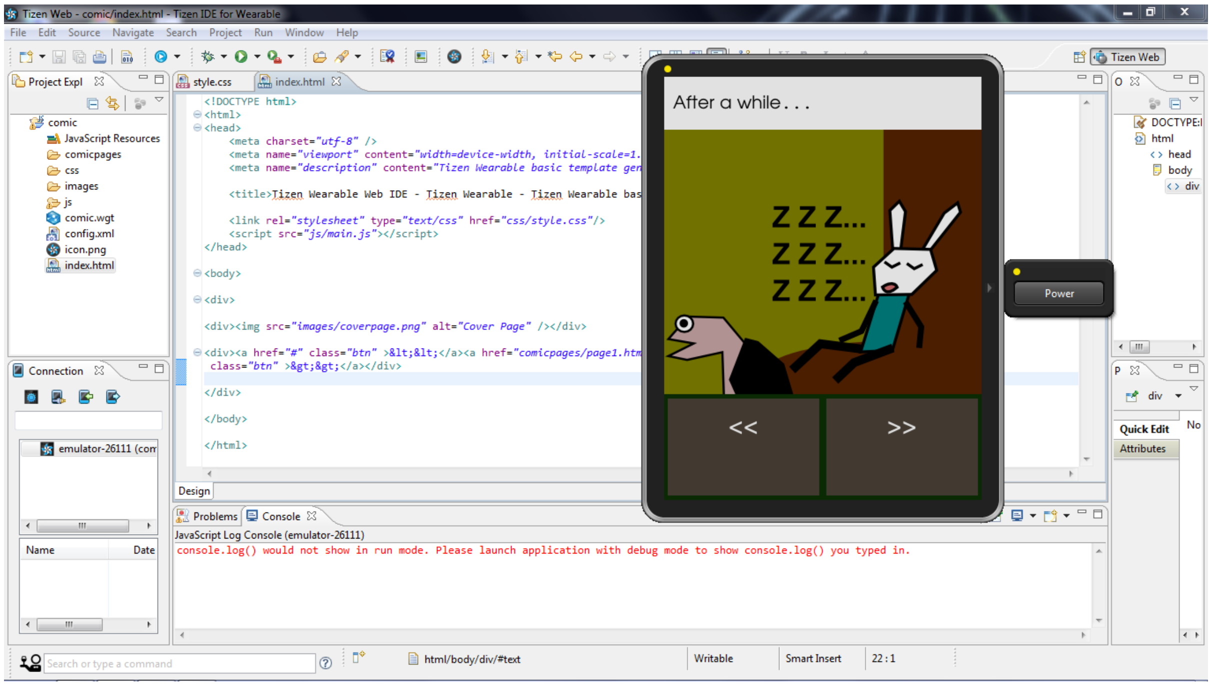
```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

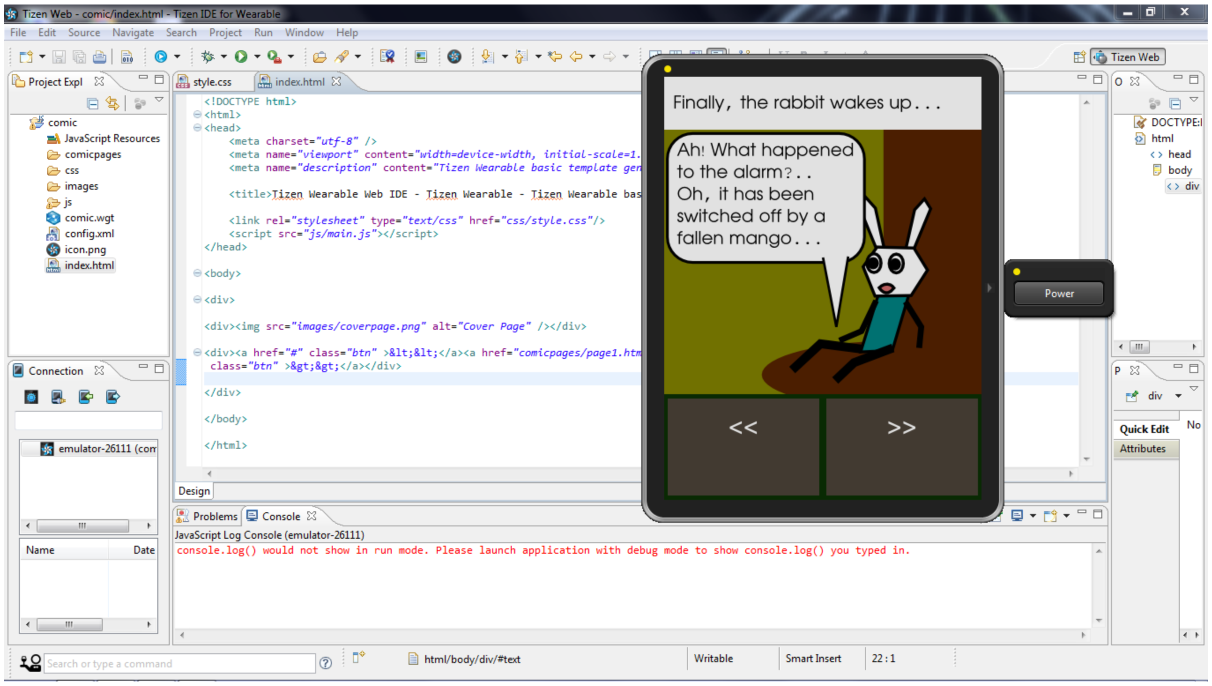
```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

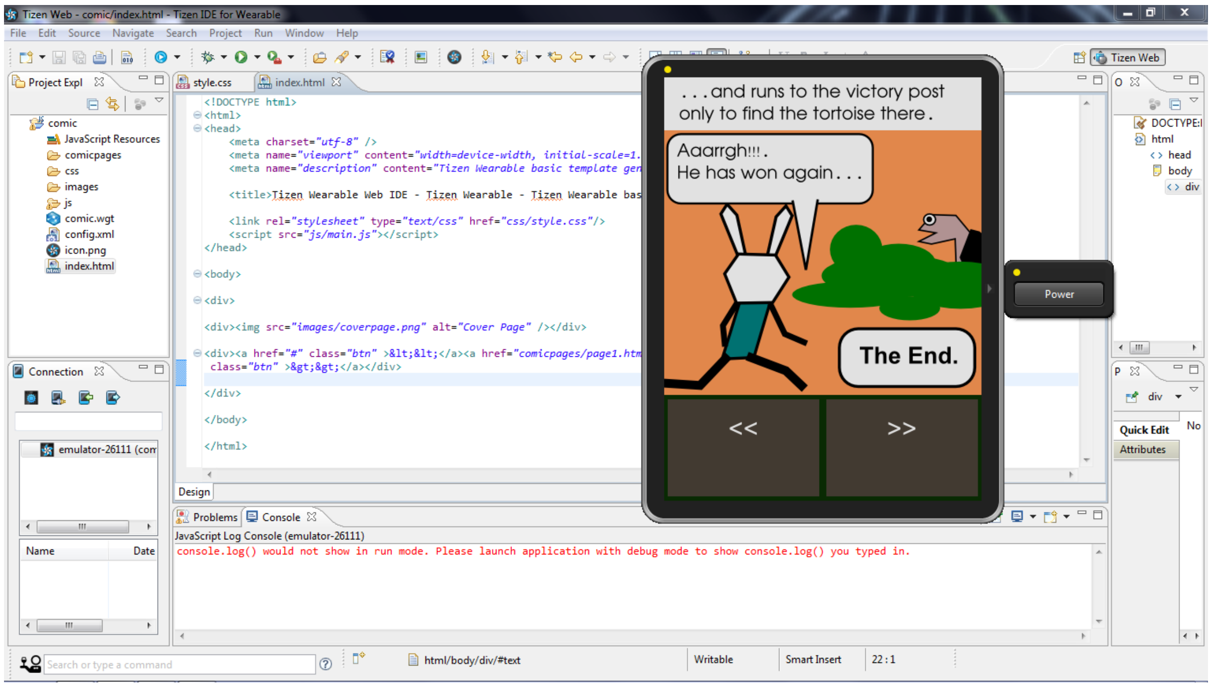
```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <meta name="description" content="Tizen Wearable basic template gen

    <title>Tizen Wearable Web IDE - Tizen Wearable - Tizen Wearable bas

    <link rel="stylesheet" type="text/css" href="css/style.css"/>
    <script src="js/main.js"></script>
</head>

<body>

<div>

<div><img src="images/coverpage.png" alt="Cover Page" /></div>

<div><a href="#" class="btn" >&lt;&lt;</a><a href="comicpages/page1.htm
 class="btn" >&gt;&gt;</a></div>

</div>

</body>

</html>
```

JavaScript Log Console (emulator-26111)

```
console.log() would not show in run mode. Please launch application with debug mode to show console.log() you typed in.
```

html/body/div/#text

Conclusions: We built a simple comic app with the Tizen SDK for Wearable and ran it on the smart watch emulator that comes bundled with the IDE.