

Dr. D. Y. Patil Technical Campus
Dr. D. Y. Patil School of Engineering
Dr. D. Y. Patil Knowledge City, Charholi (Bk), Lohegaon, Pune – 412 105
Department of Computer Engineering



Laboratory Practice-II Lab
(T. E. Computer) 2019 Course

LABORATORY MANUAL
(Version 1, w. e. f. Jan 2022)

Prepared By
Dr. Sunil Rathod
Prof. Komal Jakotiya
(Asst. Professor)

Approved By
Dr. P. M. Agarkar
(H. O. D.)

Dr. D. Y. Patil School of Engineering, Pune -412 105

Department of Computer Engineering		
Class: T.E	Laboratory Practice II (LP-II)	Semester: VI
Practical Plan		
Practical hours per week: 04		
Notes : 1. Part-I (AI) & Part-II (Cloud Computing) All assignments are compulsory. 2. Operating System recommended :- 64-bit Open source Linux or its derivative 3. Programming tools recommended: - Open Source Programming tool like G++/GCC /Java/Python/Sharp/Apex		

Sr . No.	As sign No .	Planned Date	BROAD TOPICS TO BE COVERED	Name of Experiment	Actual Date of Conduction	Remark
1.	1	20/01/22		Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.		
2.	2	27/01/22		Implement A star Algorithm for any game search problem.		
3.	3	03/02/22		Implement Greedy search algorithm for any of the following application: I. Selection Sort II. Minimum Spanning Tree III. Single-Source Shortest Path Problem IV. Job Scheduling Problem V. Prim's Minimal Spanning Tree Algorithm VI. Kruskal's Minimal Spanning Tree Algorithm VII. Dijkstra's Minimal Spanning Tree Algorithm		
4.	4	10/02/22		Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph colouring problem		

5.	5	17/02/22		Develop an elementary chatbot for any suitable customer interaction application.		
6.	6	24/02/22		Implement any one of the following Expert System I. Information management II. Hospitals and medical facilities III. Help desks management IV. Employee performance evaluation V. Stock market trading VI. Airline scheduling & cargo schedules		
7.	7	03/03/22		Case study on Amazon EC2 and learn about Amazon EC2 web services.		
8.	8	10/03/22		Installation and configure Google App Engine.		
9.	9	17/03/22		Creating an Application in Salesforce.com using Apex programming Language.		
10.	10	24/03/22		Design and develop custom Application (Mini Project) using Salesforce Cloud		
11.	11	31/03/22		Mini-Project Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open-source technologies to implement with HDFS . Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form.		

Experiment No:A1

Title: Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Aim:
Implement DFS & BFS in Artificial Intelligence.

Prerequisites:

- Concept of DFS & BFS studied in Data Structure.

Objectives:

- Student should be able to implement DFS & BFS using AI concepts.

Theory:

The **Depth First Search (DFS)** is a search technique which searches to the *lowest depth* or *ply* of the tree. The **DFS** uses *Stack* as a data structure (**OPEN** list) to process the given state space.

Example: Graph

$V = \{a, b, c, d\}$

$E = \{a-b, a-c, b-c, b-d, c-d, \}$

Start = {a}

Goal = {d}

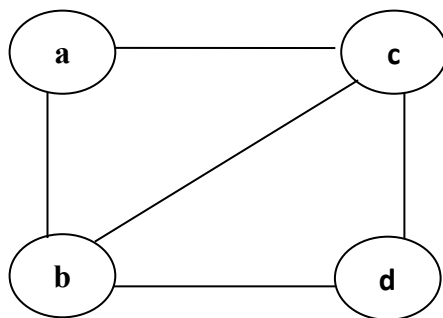


Figure-1.1: Graph

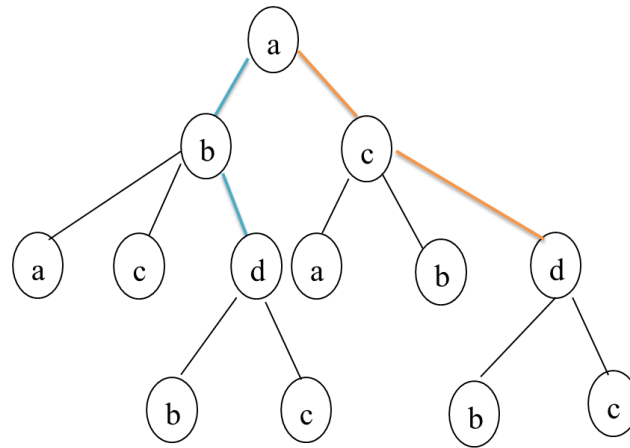


Fig-1.2: State Space Representation

$S = \{ a:[b,c], b:[a,c,d], c:[a,b,d], d:[b,c] \}$

Algorithm DFS()

Def **DFS**(Start)

Open=Start

Closed={ }

State = FAILURE

While (Open \neq Empty) AND (State \neq SUCCESS) then

 Pick front node 'N' from Open

 Open = Open - {N}

 If **GOALTEST**(N) = TRUE then

 State = SUCCESS

 Closed = **APPEND** (Closed , {N})

 Else

 Closed = **APPEND** (Closed , {N})

 Child = { **MOVEGEN**(N) }

 Child = { Child - Open }

 Child = { Child - Closed }

 Open = **APPEND** (Child,Open)

 End If

End While

Return State

Def **GOALTEST**(N)

```

If N = Goal then
  Return TRUE
Else
  Return FALSE

```

Def **MOVEGEN(N)**

Succ = { }

For N in S do

 Succ = Succ U {Children of N}

Return Succ

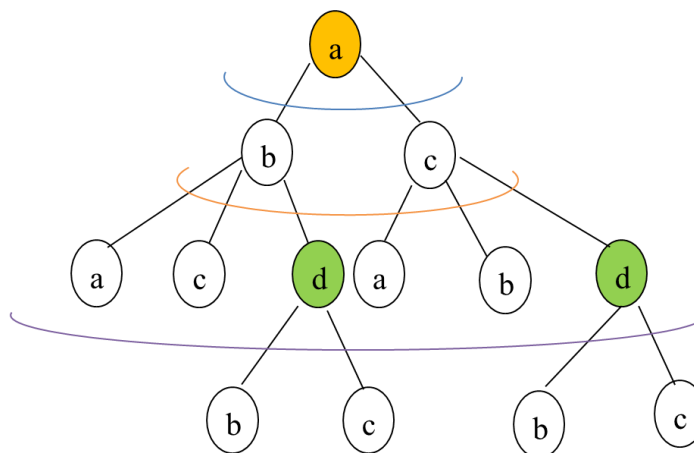
Def **APPEND(list1, list2)**

New_list = list1 + list2

Return New_list

Analysis of DFS()

1. **Completeness:** The DFS does not guarantee the solution. If the search tree has infinite depth, the algorithm may not terminate.
2. **Optimality:** The solution given by DFS is not low cost or best solution as the alternate path may be the best one.
3. **Time Complexity:** The algorithm takes exponential time. The worst case the algorithm will take time $O(b^d)$.
4. **Space Complexity:** However the space taken is linear in the depth of the search tree, $O(bN)$. Where N is the maximum depth of a node in the search space



The **Breadth First Search (BFS)** is a search technique which searches the tree *level wise* and *left to right* at each level of the tree.

The **BFS** uses *Queue* as a data structure (**OPEN** list) to process the given state space.

Algorithm BFS()Def **BFS**(Start)

Open=Start

Closed={ }

State = FAILURE

While (Open \neq Empty) AND (State \neq SUCCESS) then

Pick front node 'N' from Open

Open = Open – {N}

 If **GOALTEST**(N) = TRUE then

State = SUCCESS

 Closed = **APPEND** (Closed , {N})

Else

 Closed = **APPEND** (Closed , {N}) Child = { **MOVEGEN**(N) }

Child = { Child - Open }

Child= { Child - Closed }

 Open = **APPEND** (Open, Child)

End If

End While

Return State

Analysis of BFS()

1. **Completeness:** The BFS does guarantee the solution.
2. **Optimality:** The solution given by BFS is low cost or best solution as it inspects all child nodes at each level.
3. **Time Complexity:** The algorithm faces combinatorial explosion problem. In worst case the algorithm will take time $O(b^d)$.
4. **Space Complexity:** However the space taken is exponential, $O(b^d)$. Where d is the depth of search space tree.

Code of DFS

```
""SuccList={'a':['b','c'],'b':['a','c','d'],'c':['a','b','d'],'d':['b','c']}""
SuccList = { 'A':['B','C'], 'B':['D','E','F'], 'C':['G','H','I'], 'D':[],'E':[],'F':['J','K'], 'G':[], 'H':[], 'I':['L','M'], 'J':[],
'K':[], 'L':[], 'M':[] }
Start='A'
Goal='L'
Closed = list()
SUCCESS=True
FAILURE=False
State=FAILURE
```

```
def GOALTEST(N):
    if N == Goal:
        return True
    else:
        return False
```

```
def MOVEGEN(N):
    New_list=list()
    if N in SuccList.keys():
        New_list=SuccList[N]
    print("New_list=",New_list)
    return New_list
```

```
def APPEND(L1,L2):
    New_list=L1+L2
    return New_list
```

```
def DFS():
    OPEN=[Start]
    CLOSED=list()
    global State
    global Closed
    while (len(OPEN) != 0) and (State != SUCCESS):
        print("-----")
        N= OPEN[0]
        print("N=",N)
        del OPEN[0] #delete the node we picked

        if GOALTEST(N)==True:
            State = SUCCESS
            CLOSED = APPEND(CLOSED,list(N))
```



```

        print("CLOSED=",CLOSED)
    else:
        CLOSED = APPEND(CLOSED,list(N))
        print("CLOSED=",CLOSED)
        CHILD = MOVEGEN(N)
        print("CHILD=",CHILD)
        for val in CLOSED:
            if val in CHILD:
                CHILD.remove(val)
        for val in OPEN:
            if val in CHILD:
                CHILD.remove(val)
        OPEN = APPEND(CHILD,OPEN) #append movegen elements to OPEN
        print("OPEN=",OPEN)
    Closed=CLOSED
    return State

```

```

#Driver Code
result=DFS() #call search algorithm
print(Closed,result)

```

Code of BFS

```

SuccList={'a':['b','c'],'b':['a','c','d'],'c':['a','b','d'],'d':['b','c']}
Start='a'
Goal='d'
Closed = list()
SUCCESS=True
FAILURE=False
State=FAILURE

```

```

def GOALTEST(N):
    if N == Goal:
        return True
    else:
        return False

```

```

def MOVEGEN(N):
    New_list=list()
    if N in SuccList.keys():
        New_list=SuccList[N]

```

```
    print("New_list=",New_list)
    return New_list

def APPEND(L1,L2):
    New_list=L1+L2
    return New_list

def BFS():
    OPEN=[Start]
    CLOSED=list()
    global State
    global Closed
    while (len(OPEN) != 0) and (State != SUCCESS):
        print("-----")
        N= OPEN[0]
        print("N=",N)
        del OPEN[0] #delete the node we picked

        if GOALTEST(N)==True:
            State = SUCCESS
            CLOSED = APPEND(CLOSED,list(N))
            print("CLOSED=",CLOSED)
        else:
            CLOSED = APPEND(CLOSED,list(N))
            print("CLOSED=",CLOSED)
            CHILD = MOVEGEN(N)
            print("CHILD=",CHILD)
            for val in CLOSED:
                if val in CHILD:
                    CHILD.remove(val)
            for val in OPEN:
                if val in CHILD:
                    CHILD.remove(val)
            OPEN = APPEND(OPEN,CHILD) #append movegen elements to OPEN
            print("OPEN=",OPEN)

    Closed=CLOSED
    return State

#Driver Code
result=BFS() #call search algorithm
print(Closed,result)
```

Facilities: Fedora Operating Systems, Python

Application:

The algorithm DFS and BFS are used in the problem of graph and computer network.

Input :

The State Space representation of a given Graph.

Output :

The output is the path traversed by the algorithm from start node to goal node. If search fails to find the goal then it returns the search status as FAILURE.

Conclusion:

The implementation simulates the working of DFS and BFS on given graph in the perspective of AI.

Questions:

1. What is Search Algorithm?
2. What is DFS and BFS?
3. What are OPEN list and CLOSED list data structure and why it is used?
4. What is the significance of GOALTEST() and MOVEGEN() function?
5. What are the different parameters of analyzing the DFS and BFS.
6. What is State Space representation of a given problem.

Experiment No:A2

Title: Implement A* Algorithm for any game search problem.

Aim: Implementation of A* Algorithm for any game search problem.

Prerequisites:

Concept of Heuristic function, state space.

Objectives:

Student should be able to implement A* algorithm on given graph.

Theory:

A* (pronounced as "A star") is a computer algorithm that is widely used in path finding and graph traversal. However, the A* algorithm introduces a heuristic into a regular graph-searching algorithm, essentially planning ahead at each step so a more optimal decision is made.

A* is an extension of Dijkstra's algorithm with some characteristics of breadth-first search (BFS).

A* uses a function $f(n)$ that gives an estimate of the total cost of a path using that node. Therefore, A* is a heuristic function, which differs from an algorithm in that a heuristic is more of an estimate and is not necessarily provably correct.

A* expands paths that are already less expensive by using this function:

$$f(n)=g(n)+h(n)$$

where,

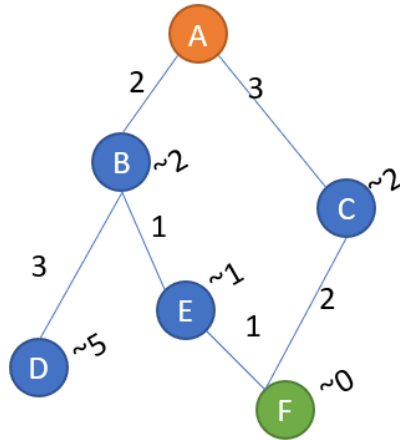
- $f(n)$ = total estimated cost of path through node n
- $g(n)$ = cost so far to reach node n
- $h(n)$ = estimated cost from n to goal. This is the heuristic part of the cost function, so it is like a guess.

The calculation of $h(n)$ can be done in various ways:

- The Manhattan distance (explained below) from node n to the goal is often used. This is a standard heuristic for a grid.
- If $h(n) = 0$, A* becomes Dijkstra's algorithm, which is guaranteed to find a shortest path.

The heuristic function must be **admissible**, which means it can never overestimate the cost to reach the goal. Both the Manhattan distance and $h(n) = 0$ are admissible.

Consider the following graph with edge cost (distance between two nodes like A-B, C-F, etc) and the heuristics value to reach to the target node.



The above graph can be represented as dictionary data structure of Python as:

```
graph={
    # to store g-score and h-score
    # list first value is the g-score, second value is the h-score,i.e., heuristic
    'A':{'B':[2,2],'C':[3,2]},
    'B':{'D':[3,5],'E':[1,1]},
    'C':{'F':[2,0]},
    'D':{},
    'E':{'F':[1,0]},
    'F':{}
}
```

The algorithm will retrieve the graph as follow:

#graph['A'] this return {'B':[2,2],'C':[3,2]}

#graph['A']['B'] this return [2,2]

#graph['A']['B'][0] return the edge length

#graph['A']['B'][1] return the distance of the node to destination

```
def astar(graph,start_node,end_node):
```

```
    # astar: F=G+H, we name F as f_distance, G as g_distance, H as heuristic
```

```
    #Assign all the nodes, a f_distance value as infinity as initial value
```

```
    f_distance={node:float('inf') for node in graph}
```

```
    #The f_ditance value of start node is 0
```

```
    f_distance[start_node]=0
```

```
    #Assign all the nodes, a g_distance value as infinity as initial value
```

```
    g_distance={node:float('inf') for node in graph}
```

```
    #The g_ditance value of start node is 0
```

```
    g_distance[start_node]=0
```

```

#Keep the track of parent node in came_form
came_from={node:None for node in graph}
came_from[start_node]=start_node

queue=[(0,start_node)] #use queue as list
while queue:
    f_distance,current_node=heapq.heappop(queue)
    if current_node == end_node:
        print('found the end_node')
        return f_distance, came_from
    #for all the neighbors of the current node calculate g_distance
    for next_node,weights in graph[current_node].items():
        temp_g_distance=g_distance[current_node]+weights[0]
        #g_distance of current node is less than the g_distance of neighbor
        #Update the g_distance of next node to the smaller distance value.
        if temp_g_distance<g_distance[next_node]:
            g_distance[next_node]=temp_g_distance
            heuristic=weights[1]
            f_distance=temp_g_distance+heuristic
            came_from[next_node]=current_node
            heapq.heappush(queue,(f_distance,next_node))

return f_distance, came_from

#Driver Code
Node_distance, Path=astar(graph,'A','F')
print(Node_distance)
print(Path)

```

8-Puzzle using A* Algorithms

N-Puzzle or **sliding puzzle** is a popular puzzle that consists of N tiles where N can be 8, 15, 24, and so on. In our example N = 8. The puzzle is divided into $\sqrt{N+1}$ rows and $\sqrt{N+1}$ columns

Eg. 15-Puzzle will have 4 rows and 4 columns and an 8-Puzzle will have 3 rows and 3 columns. The puzzle consists of N tiles and one empty space where the tiles can be moved. Start and Goal configurations (also called state) of the puzzle are provided.

Initial State			Goal State		
1	2	3	2	8	1
8		4		4	3
7	6	5	7	6	5

Rules for solving the puzzle.

Instead of moving the tiles in the empty space, we can visualize moving the empty space in place of the tile, basically swapping the tile with the empty space. The empty space can only move in four directions viz.

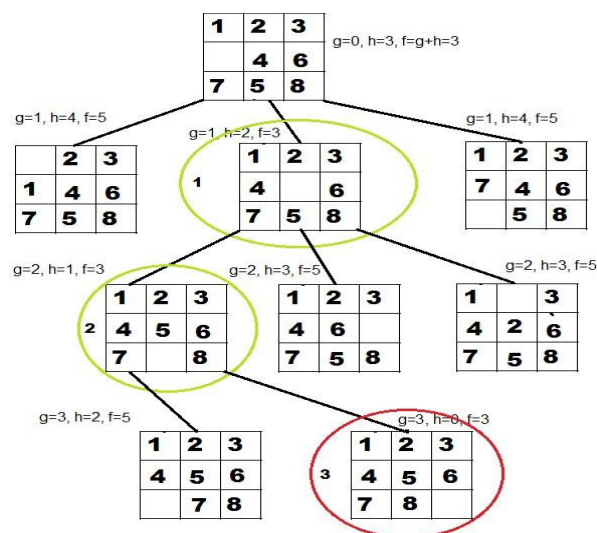
- 1.Up
- 2.Down
- 3.Right
4. Left

The empty space cannot move diagonally and can take only one step at a time.

In our 8-Puzzle problem, we can define the **h-score** as the number of misplaced tiles by comparing the current state and the goal state or summation of the Manhattan distance between misplaced nodes. **g-score** will remain as the number of nodes traversed from a start node to get to the current node.

From Figure below, we can calculate the **h-score** by comparing the initial(current) state and goal state and counting the number of misplaced tiles.

Thus, **h-score** = 5 and **g-score** = 0 as the number of nodes traversed from the start node to the current node is 0.



How A* solves the 8-Puzzle problem.

We first move the empty space in all the possible directions in the start state and calculate the f-score for each state.

This is called expanding the current state.

After expanding the current state, it is pushed into the **closed** list and the newly generated states are pushed into the **open** list. A state with the least f-score is selected and expanded again. This process continues until the goal state occurs as the current state. Basically, here we are providing the algorithm a measure to choose its actions. The algorithm chooses the best possible action and proceeds in that path. This solves the issue of generating redundant child states, as the algorithm will expand the node with the least **f-score**.

Code:

```
def accept(n):
    """ Accepts the puzzle from the user """
    puz = []
    for i in range(n):
        puz.append([val for val in input().split()])
    return puz

def print_board(board,n):
    for i in range(n):
        print()
        for j in range(n):
            print(board[i][j],end=' ')

#Find the position of blank space
def find_space(Current,n):
    for blank_row_pos in range(n):
        for blank_col_pos in range(n):
            if Current[blank_row_pos][blank_col_pos]=='_':
                return blank_row_pos,blank_col_pos

#Copy the current node to new node for shuffling the blank space and create a new configuration
def copy_current(Current):
    temp=[]
    for i in range(len(Current)):
        row=[]
        for val in Current[i]:
            row.append(val)

        temp.append(row)

    return(temp)

#Move the blank space in given direction, if out of range return None
```



```

def shuffle(Current,brow_pos,bcol_pos,move_x,move_y):
    if move_x >= 0 and move_x < len(Current) and move_y >= 0 and move_y < len(Current):
        temp=[]
        temp=copy_current(Current)
        change=temp[move_x][move_y]
        temp[move_x][move_y]=temp[brow_pos][bcol_pos]
        temp[brow_pos][bcol_pos]=change
        return temp
    else:
        return None

#Function to calculate g_score: the number of nodes traversed from a start node to get to the current node
def g_score(Node):

    return Node[1] #Node=[Board,level,fscore]

#Function to calculate h_score: the number of misplaced tiles by comparing the current state and the goal
state
def h_score(Current,Goal,n):
    hscore=0
    for i in range(n):
        for j in range(n):
            if (Current[i][j] != Goal[i][j]) and (Current[i][j]!='_'):
                hscore +=1

    return hscore

#Function to calculate f_Score= g_score + h_Score
def f_score(Node,Goal,n):
    Current=Node[0]
    return g_score(Node) + h_score(Current,Goal,n)

#Generate the child nodes by moving the blank in any four direction (up,down,left,right)
def move_gen(Node,Goal,n):
    Current=Node[0]
    level=Node[1]
    fscore=0
    row,col=find_space(Current,n)
    move_positions=[[row,col-1],[row,col+1],[row-1,col],[row+1,col]] #left,right,up,down

    children=[] #List of child nodes of current node

    for move in move_positions:
        child=shuffle(Current,row,col,move[0],move[1])
        if child is not None:
            cNode=[child,0,0] #Dummy node for calculating f_Score
            fscore=f_score(cNode,Goal,n)

```

```

        Node=[child,level+1,fscore]
        children.append(Node)
    print("\n\n The Children ::",children)
    return children

#Function goal_test to see the goal configuration is reached
def goal_test(Current,Goal,n):
    if h_score(Current,Goal,n) == 0:
        return True
    else:
        return False

#Function to Sort OPEN based on f_score
def sort(L):
    L.sort(key = lambda x: x[2],reverse=False)
    return L

#Function for starting the Game
def play_game(Start, Goal, n):
    #when game starts
    fscore=0 #fscore initialized to zero
    gscore=0 #gscore initialized to zero
    level=0 #the start configuration is root node s at level-0 of the state space tree
    ""
    print_board(Start,n);
    print_board(Goal,n)
    print("\n\nI AM HERE !!!\n\n")
    ""
    Node=[Start,level,fscore]
    fscore=f_score(Node,Goal,n)

    #Every Node is [board configuration ,level,gscore]
    Node = [Start,level,fscore] # current node is Start node
    print("\nThe Node is=\n",Node)
    OPEN = [] #OPEN list as frontier
    CLOSED = [] #CLOSED as explored
    OPEN.append(Node)
    levelcount=0

    #Explored the current node to reach to the Goal configuration
    while True:

        N=OPEN[0] #first node of open
        del OPEN[0] # delete first node of open

        Current=N[0] #Extract board configuration
        print("\n\n The current configuration is ::",Current)

```

```

CLOSED.append(N)
#if goal configuration is reached terminate
if goal_test(Current,Goal,n) == True:
    print("\nGoal reached!!")
    print("CLOSED=",CLOSED)
    break

CHILD=move_gen(N,Goal,n)
#print("\n\n The CHILD is ::",CHILD)
OPEN=[]
for child in CHILD:
    OPEN.append(child)
#sort the OPEN list based on fscore value of each node
sort(OPEN)
#print("\n\n The OPEN is ::",OPEN)

```

```

#Drive Code
n=int(input("Enter the board size:"))

print("\nEnter Start Configuration of board")
Start=accept(n)

print("\nEnter Goal Configuration of board")
Goal=accept(n)

play_game(Start, Goal, n)

```

Facilities: Fedora Operating Systems, Python

Application:

A* Algorithm is used in various diversified areas of research, computer network & graph, Google maps to find the shortest path between source and destination.

Input :

State Space representation of graph.

Output :

The shortest path between source and destination.

Conclusion:

The A* algorithm finds the shortest path between source and destination using heuristic function.

Questions:

1. What is AO graph?
2. What is AO algorithm?
3. What is A* algorithm and how it is an AI based implementation of search problem?
4. What is significance of “Star” (*) in the A* algorithm?

Experiment No:A3

Title: Implement Greedy search algorithm for any of the following application:

- I. Selection Sort
- II. Minimum Spanning Tree
- III. Single-Source Shortest Path Problem
- IV. Job Scheduling Problem
- V. Prim's Minimal Spanning Tree Algorithm
- VI. Kruskal's Minimal Spanning Tree Algorithm
- VII. Dijkstra's Minimal Spanning Tree Algorithm

Aim: Implement Dijkstra's Minimal Spanning Tree Algorithm

Prerequisites:

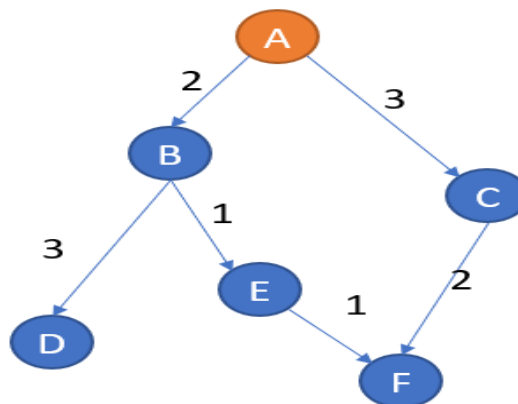
- Concept of **Graph** and the Graph representation using Adjacency Matrix/List.

Objectives:

Student should be able to implement Dijkstra's algorithm

Theory:

Consider the following graph given in figure below.



The state space representation of the above graph is done using dictionary data structure as follows.

```

graph={
  'A': {'B':2, 'C':3},
  'B': {'D':3, 'E':1},
  'C': {'F':2},
  'D': {},
  'E': {'F':1},
  'F': {}
}

```

The above graph representation is a python dictionary wherein a node and its immediate successors or child nodes are specified, for example, Node 'A' has node 'B' and node 'C' as successors. The path cost from node 'A' to node 'B' is 2 and that of path 'A' to 'C' is 3.

Dijkstra's algorithm:

function Dijkstra(*Graph*, *source*):

$\text{dist}[\text{source}] \leftarrow 0$ // the start node has distance 0
 create vertex priority queue *Q* //Use priority Queue here

for each vertex *v* in *Graph*:

 if $v \neq \text{source}$

$\text{dist}[v] \leftarrow \text{INFINITY}$ //Assign the weight

$\text{prev}[v] \leftarrow \text{UNDEFINED}$ //predecessor of current node *v*

$Q.\text{add_with_priority}(v, \text{dist}[v])$ //

while *Q* is not empty:

$u \leftarrow Q.\text{extract_min}()$ // Remove and return best vertex

for each neighbor *v* of *u* still in *Q*: // only *v* that are still in *Q*

$\text{alt} \leftarrow \text{dist}[u] + \text{length}(u, v)$

if $\text{alt} < \text{dist}[v]$:

$\text{dist}[v] \leftarrow \text{alt}$

$\text{prev}[v] \leftarrow u$

$Q.\text{decrease_priority}(v, \text{alt})$

return *dist*, *prev*

Code of Dijkstra's Algorithm:

```
import heapq
```

```
graph={
```

```
    'A':{'B':2,'C':3},
```

```
    'B':{'D':3,'E':1},
```

```
    'C':{'F':2},
```

```
    'D':{},
```

```
    'E':{'F':1},
```

```
    'F':{}
}
```

```
def dijkstra(graph,node):
```

```
    #The distance value of start node is zero '0'
```

```
    distances[node]=0
```

```
    #Assign infinity to all other nodes
```

```
    distances={node:float('inf') for node in graph}
```

```
    print("distances::",distances)
```

```
    #Predecessor of node is stored here
```

```
    previous={node:None for node in graph}
```

```

queue=[(0,node)] #queue stores start 'node' with edge distance value '0'.

while queue:
    #heapq of python maintains the priority queue (min queue)
    #heappop() method of heapq will pop the minimum val of the heap
    current_distance, current_node = heapq.heappop(queue)

    # relaxation, visit all the successors of current_node and get the edge cost as well
    for next_node, weight in graph[current_node].items():
        distance_temp=current_distance+weight
        #if the distance of the currently visited node is smaller than the earlier stored cost #then
        update the distance value of current node with smaller cost
        if distance_temp<distances[next_node]:
            distances[next_node]=distance_temp
            previous[next_node]=current_node
            heapq.heappush(queue,(distance_temp,next_node))
        print("Distances::",distances)
    return distances,previous

#Driver Code
Node_distance, Path = dijkstra(graph,'A')
print(Node_distance)
print(Path)

```

Facilities: Fedora Operating Systems, Python

Application:

The algorithm finds the Minimal Spanning Tree for given graph when a source is specified so it is used in Google Map and other similar application to find the shortest route between the source and destination.

Input:

State Space representation of the given Graph or the problem.

Output:

The algorithm finds the Minimal Spanning Tree for given graph when a source is specified.

Conclusion:

The Dijkstra's Algorithm finds the Single source and multiple destinations with shortest distance between the two nodes in the given graph thereby finding the shortest distance between a given source and the destination.

Questions:

1. What is Minimal Spanning Tree?
2. What is shortest path algorithm?

3. What is single source multiple destination problem?
4. What are multiple sources and multiple destination problems?

Experiment No: 4

Title: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

Aim:

Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for N-queens problem.

Prerequisites:

Concept of CSP, backtracking and branch & bound technique.

Objectives:

Student should be able to implement the 8-Queen problem using CSP, backtracking and branch & bound technique.

Theory:

A constraint satisfaction problem consists of three components, X, D , and C :

X is a set of variables, $\{X_1, \dots, X_n\}$.

D is a set of domains, $\{D_1, \dots, D_n\}$, one for each variable.

C is a set of constraints that specify allowable combinations of values.

Each domain D_i consists of a set of allowable values, $\{v_1, \dots, v_k\}$ for variable X_i . Each constraint C_i consists of a pair $\langle \text{scope}, \text{rel} \rangle$, where scope is a tuple of variables that participate in the constraint and rel is a relation that defines the values that those variables can take on. A relation can be represented as an explicit list of all tuples of values that satisfy the constraint, or as an abstract relation that supports two operations: testing if a tuple is a member of the relation and enumerating the members of the relation. For example, if X_1 and X_2 both have the domain $\{A, B\}$, then the constraint saying the two variables must have different values can be written as $\langle (X_1, X_2), [(A, B), (B, A)] \rangle$ or as $\langle (X_1, X_2), X_1 \neq X_2 \rangle$.

To solve a CSP, we need to define a state space and the notion of a solution. Each state in a CSP is defined by an **assignment** of values ASSIGNMENT to some or all of the variables, $\{X_i = v_i, X_j = v_j, \dots\}$. An assignment that does not violate any constraints is called a **consistent** or legal assignment. A **complete assignment** is one in which every variable is assigned, and a **solution** to a CSP is a consistent, complete assignment. A **partial assignment** is one that assigns values to only some of the variables.

The N-Queens problem is a puzzle of placing exactly N queens on an $N \times N$ chessboard, such that no two queens can attack each other in that configuration. Thus, no two queens can lie in the same row, column or diagonal.

In the backtracking algorithm, we consider possible configurations one by one and backtrack if we hit a dead end.

The branch and bound solution is somehow different, it generates a partial solution until it figures that there's no point going deeper as we would ultimately lead to a dead end.

In the backtracking approach, we maintain an 8×8 binary matrix for keeping track of safe cells (by eliminating the unsafe cells, those that are likely to be attacked) and update it each time we place a new queen. However, it required $O(n^2)$ time to check safe cell and update the queen.

In the 8 queens problem, we ensure the following:

1. no two queens share a row
2. no two queens share a column
3. no two queens share the same left diagonal
4. no two queens share the same right diagonal

we already ensure that the queens do not share the same column by the way we fill out our auxiliary matrix (column by column). Hence, only the left out 3 conditions are left out to be satisfied.

Applying the backtracking approach:

Now, we place queen q_1 in the very first acceptable position (1, 1). Next, we put queen q_2 so that both these queens do not attack each other. We find that if we place q_2 in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for q_2 in column 3, i.e. (2, 3) but then no position is left for placing queen ' q_3 ' safely. So we backtrack one step and place the queen ' q_2 ' in (2, 4), the next best possible solution. Then we obtain the position for placing ' q_3 ' which is (3, 2). But later this position also leads to a dead end, and no place is found where ' q_4 ' can be placed safely. Then we have to backtrack till ' q_1 ' and place it to (1, 2) and then all other queens are placed safely by moving q_2 to (2, 4), q_3 to (3, 1) and q_4 to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.

	1	2	3	4
1			q_1	
2	q_2			
3				q_3
4		q_4		

The implicit tree for 4 - queen problem for a solution (2, 4, 1, 3) is as follows:

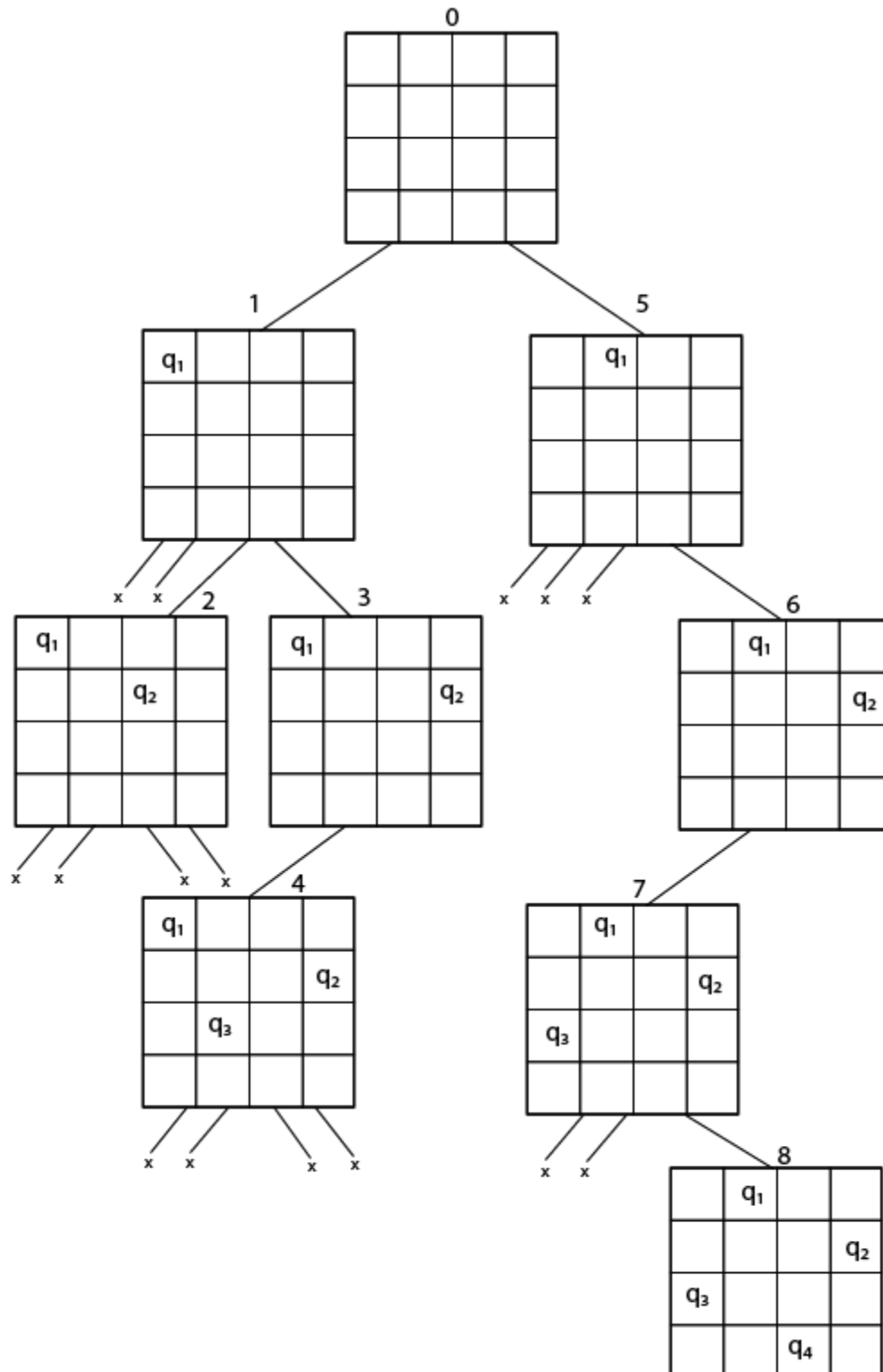
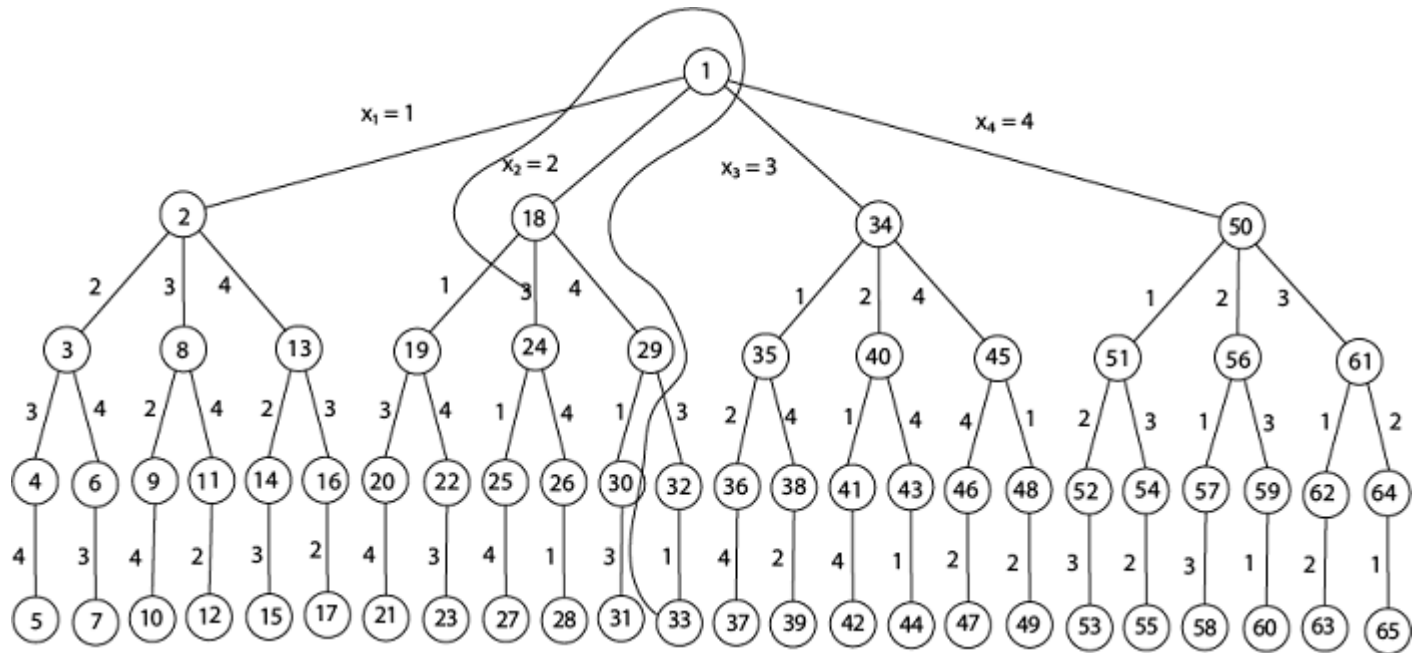


Fig shows the complete state space for 4 - queens problem. But we can use backtracking method to generate the necessary node and stop if the next node violates the rule, i.e., if two queens are attacking.



4 - Queens solution space with nodes numbered in DFS

It can be seen that all the solutions to the 4 queens problem can be represented as 4 - tuples (x_1, x_2, x_3, x_4) where x_i represents the column on which queen " q_i " is placed.

One possible solution for 8 queens problem is shown in fig:

	1	2	3	4	5	6	7	8
1				q_1				
2						q_2		
3								q_3
4		q_4						
5							q_5	
6	q_6							
7			q_7					
8					q_8			

1. Thus, the solution for 8 -queen problem for (4, 6, 8, 2, 7, 1, 3, 5).
2. If two queens are placed at position (i, j) and (k, l).
3. Then they are on same diagonal only if $(i - j) = k - l$ or $i + j = k + l$.
4. The first equation implies that $j - l = i - k$.
5. The second equation implies that $j - l = k - i$.
6. Therefore, two queens lie on the duplicate diagonal if and only if $|j-l|=|i-k|$

Python3 program to solve N Queen

Problem using backtracking

global N

N = 4

def printSolution(board):

 for i in range(N):

 for j in range(N):

 print (board[i][j], end = " ")

 print()

A utility function to check if a queen can

be placed on board[row][col]. Note that this

function is called when "col" queens are

already placed in columns from 0 to col -1.

So we need to check only left side for

attacking queens

def isSafe(board, row, col):

 # Check this row on left side

 for i in range(col):

 if board[row][i] == 1:

```
        return False

    # Check upper diagonal on left side
    for i, j in zip(range(row, -1, -1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check lower diagonal on left side
    for i, j in zip(range(row, N, 1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQUtil(board, col):

    # base case: If all queens are placed
    # then return true
    if col >= N:
        return True

    # Consider this column and try placing
    # this queen in all rows one by one
    for i in range(N):

        if isSafe(board, i, col):
```

```
# Place this queen in board[i][col]
board[i][col] = 1

# recur to place rest of the queens
if solveNQUtil(board, col + 1) == True:
    return True

# If placing queen in board[i][col]
# doesn't lead to a solution, then
# queen from board[i][col]
board[i][col] = 0

# if the queen can not be placed in any row in
# this column col then return false
return False

# This function solves the N Queen problem using
# Backtracking. It mainly uses solveNQUtil() to
# solve the problem. It returns false if queens
# cannot be placed, otherwise return true and
# placement of queens in the form of 1s.
# note that there may be more than one
# solutions, this function prints one of the
# feasible solutions.
def solveNQ():
    board = [ [0, 0, 0, 0],
               [0, 0, 0, 0],
               [0, 0, 0, 0],
               [0, 0, 0, 0] ]
```

```
if solveNQUtil(board, 0) == False:
    print ("Solution does not exist")
    return False

printSolution(board)
return True
```

Driver Code

```
solveNQ()
```

Applying the branch and bound approach:

The branch and bound approach suggests that we create a partial solution and use it to ascertain whether we need to continue in a particular direction or not. For this problem, we create 3 arrays to check for conditions 1, 3 and 4. The boolean arrays tell which rows and diagonals are already occupied. To achieve this, we need a numbering system to specify which queen is placed.

The indexes on these arrays would help us know which queen we are analyzing.

Preprocessing - create two $N \times N$ matrices, one for top-left to bottom-right diagonal, and other for top-right to bottom-left diagonal. We need to fill these in such a way that two queens sharing same top-left_bottom-right diagonal will have same value in slashDiagonal and two queens sharing same top-right_bottom-left diagonal will have same value in backSlashDiagonal.

$\text{slashDiagonal}(\text{row})(\text{col}) = \text{row} + \text{col}$

$\text{backSlashDiagonal}(\text{row})(\text{col}) = \text{row} - \text{col} + (N-1) \quad \{ N = 8 \}$

{ we added (N-1) as we do not need negative values in backSlashDiagonal }

7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
11	10	9	8	7	6	5	4
12	11	10	9	8	7	6	5
13	12	11	10	9	8	7	6
14	13	12	11	10	9	8	7

slash diagnol[row][col] = row + col

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

backslash diagnol[row][col] = row-col+(N-1)

For placing a queen i on row j , check the following:

1. whether row ' j ' is used or not
2. whether slashDiagnol ' $i+j$ ' is used or not
3. whether backSlashDiagnol ' $i-j+7$ ' is used or not

If the answer to any one of the following is true, we try another location for queen i on row j , mark the row and diagonls; and recur for queen $i+1$.

""" A utility function to print solution """

```
def printSolution(board):
```

```
    for i in range(N):
```

```
        for j in range(N):
```

```
            print(board[i][j], end = " ")
```

```
        print()
```

""" A Optimized function to check if

a queen can be placed on board[row][col] """

```
def isSafe(row, col, slashCode, backslashCode,
```

```

        rowLookup, slashCodeLookup,
            backslashCodeLookup):
    if (slashCodeLookup[slashCode[row][col]] or
        backslashCodeLookup[backslashCode[row][col]] or
            rowLookup[row]):
        return False
    return True

```

""" A recursive utility function

to solve N Queen problem """

```

def solveNQueensUtil(board, col, slashCode, backslashCode,
                    rowLookup, slashCodeLookup,
                    backslashCodeLookup):

```

""" base case: If all queens are
placed then return True """

if(col >= N):

return True

for i in range(N):

if(isSafe(i, col, slashCode, backslashCode,

rowLookup, slashCodeLookup,

backslashCodeLookup)):

""" Place this queen in board[i][col] """

board[i][col] = 1

rowLookup[i] = True

slashCodeLookup[slashCode[i][col]] = True

backslashCodeLookup[backslashCode[i][col]] = True

""" recur to place rest of the queens """

```

        if(solveNQueensUtil(board, col + 1,
                                slashCode, backslashCode,
                                rowLookup, slashCodeLookup,
                                backslashCodeLookup)):
            return True

```

```

        """ If placing queen in board[i][col]
        doesn't lead to a solution,then backtrack """

```

```

        """ Remove queen from board[i][col] """
        board[i][col] = 0
        rowLookup[i] = False
        slashCodeLookup[slashCode[i][col]] = False
        backslashCodeLookup[backslashCode[i][col]] = False

```

```

        """ If queen can not be place in any row in
        this column col then return False """
        return False

```

""" This function solves the N Queen problem using
 Branch or Bound. It mainly uses solveNQueensUtil()to
 solve the problem. It returns False if queens
 cannot be placed,otherwise return True or
 prints placement of queens in the form of 1s.
 Please note that there may be more than one
 solutions,this function prints one of the
 feasible solutions. """

```

def solveNQueens():

```

```

    board = [[0 for i in range(N)]

```

```
        for j in range(N)]

# helper matrices
slashCode = [[0 for i in range(N)]
              for j in range(N)]
backslashCode = [[0 for i in range(N)]
                 for j in range(N)]

# arrays to tell us which rows are occupied
rowLookup = [False] * N

# keep two arrays to tell us
# which diagonals are occupied
x = 2 * N - 1
slashCodeLookup = [False] * x
backslashCodeLookup = [False] * x

# initialize helper matrices
for rr in range(N):
    for cc in range(N):
        slashCode[rr][cc] = rr + cc
        backslashCode[rr][cc] = rr - cc + 7

if(solveNQueensUtil(board, 0, slashCode, backslashCode,
                    rowLookup, slashCodeLookup,
                    backslashCodeLookup) == False):

    print("Solution does not exist")
    return False
```

```
# solution found  
printSolution(board)  
return True
```

```
# Driver Cde  
solveNQueens()
```

Facilities: Fedora Operating Systems, Python

Application:

The CSP, Backtracking and Branch & Bound technique are used in many applications like Map coloring, optimization problems and network programming.

Input:

Board with N x N size.

Output:

The algorithm places all the N Queens in N different columns with the given constraints.

Conclusion:

The N queen problem is implemented using python language with two techniques called Backtracking and branch & bound.

Questions:

1. What is Backtracking?
2. What is Branch and Bound technique?
3. What is CSP?
4. What is state space for 8 queen problem?

Experiment No: 5

Title: Implement the Expert System for Medical Diagnosis of 5-10 diseases based on adequate symptoms.

Aim:

Implement Expert System using prolog: Medical Diagnosis of 5-10 diseases based on adequate symptoms.

Prerequisites:

Expert System Design, Prolog, First Order Logic concepts.

Objectives:

Student should be able to implement the Expert System for Medical Diagnosis.

Theory:

What are Expert Systems?

The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

Characteristics of Expert Systems

- High performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems

The expert systems are capable of –

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem
- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Components of Expert System

The components of ES include –

- Knowledge Base

- Inference Engine
- User Interface

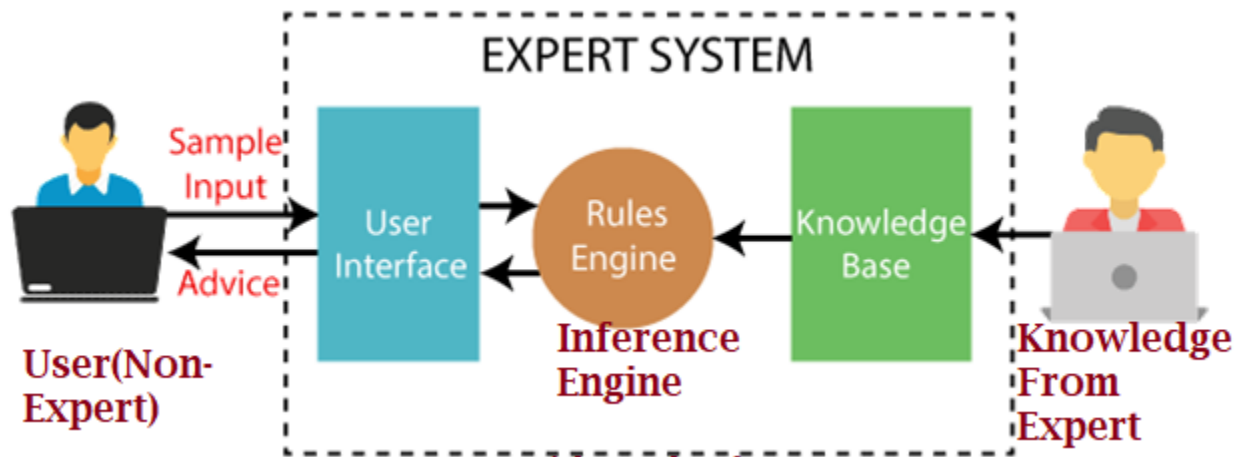


Figure:

Implementation of AI Enabled Expert System Using Prolog Platform

Knowledge Base

It contains domain-specific and high-quality knowledge. Knowledge is required to exhibit intelligence. The success of any Expert System majorly depends upon the collection of highly accurate and precise knowledge

Components of the Knowledge Base

The ES knowledge base is your store for both, factual and technical information.

True Information – It is information that is widely accepted by Information Engineers and experts in the field of work.

Heuristic Knowledge – It is about practice, accurate judgment, human ability to analyze, and guess.

Information representation

It is a method used to organize and legitimize information. It is in the form of IF-THEN-ELSE rules.

Acquisition of information

- The success of any professional program or expert system depends to a large extent on the quality, completeness, and accuracy of the information stored in the database.
- The knowledge base is built on the study of various experts, scholars, and information engineers. An information engineer is a person with the qualities of empathy, rapid learning, and case analysis skills.
- Knowledge Engineer gets information from a course specialist by recording, interviewing, and supervising him at work, etc.
- He then classifies and organizes the information in a logical manner, in accordance with the IF-THEN-ELSE rules, which will be used by the interference machine. The information engineer also monitors the development of ES.

Inference engine

The application of effective procedures and rules by Inference Engine is essential for taking the right, flawless solution.

In the event that ES is based on information, the Inference Engine acquires and uses the information available at the information center to achieve a specific solution.

In the event of Rule Based Expert System, It applies: –

- It applies repetitive rules to facts, which are found in the application of the previous law.
- Add new information to the database if needed.
- It resolves conflicts of law where many rules apply in a particular case.

To recommend a solution, the Inference Engine uses the following strategies

1] **Forward Chaining**–To recommend a solution, the Inference Engine uses the following strategies

2] **Backward Chaining**– begins with a goal and then works backwards to find out which facts need to be emphasized in order to achieve the goal.

User Interface

With the help of the user interface, the professional system works with the user, captures queries such as in readable formats, and transmits them to the inference engine.

Example of World Old Expert System-Mycin

- **MYCIN(ES):**
MYCIN was a built using backward chaining methodology who used artificial intelligence to identify bacteria that cause serious infections, such as bacterium and meningitis, and recommend antibiotics, with a dose determined by the patient's body weight – a term derived from antibiotics itself, such as anti-bacterial drugs .mycin “.
- Mycin system was also used to diagnose blood clotting disorders.
- MYCIN was developed over five or six years in the early 1970's at Stanford University.
- It was written in Lisp as Edward Shortliffe's doctoral degree directed by Bruce G. Buchanan, Stanley N. Cohen and others.

What is Prolog Language or Platform

- Prolog long form stands for programming in Logic. It is logical & declarative language that plays important role in Artificial intelligence.
- In prolog, logic is expressed as relations which is in the form of “Facts” and “Rules”. Main part in implementation of Prolog is logic being applied.
- In prolog Computation is carried out by running a query over these wriyyen relations whereas relations are formed with the help of Facts & Rules.
- In Declarative language the programmer specifies a goal to be achieved & prolog system works on how to achieve it.
- In prolog .pl extension is used for program.

Comparing Prolog with Traditional Language

- Traditional programming languages or old PL are said to be procedure oriented programming language where programmer specifies in details how to solve problems.
- Related to purely declarative language programmer only specifies what problem is & leaves rest of the part to language system itself (prolog).
- Prolog is considered to be Non-procedural language system.
- It is based on pattern matching language consist of series of Rules & Facts.

Important Features & Signs Used In Prolog

English	Calculus	Prolog
and	\wedge	,
or	\vee	;
if	\longrightarrow	$:-$
not	\sim	not

Important Features & Signs Used In Prolog

- . Indicates Prolog statement termination.
- = Indicates Non equality operators.
- ? Indicates query start point
- Facts & predicates always start with alphabet.

Terminologies Used In Prolog

Fact

- Fact is explicit relationship between object and properties of object
- In prolog, We declare some facts. These facts constitute the Knowledge Base of the expert system. We can ask query against the Knowledge Base. We can get output as affirmative if our query is already present in the knowledge Base or it is implied by Knowledge Base, otherwise we get output as negative.
- So, Knowledge Base can be considered as similar to database, against which we can ask query.
- Prolog facts are represented in definite pattern.
- Facts consist of entities and their relation.
- Entities are mentioned or written within the parenthesis separated by comma (,).
- Their relation is expressed at the beginning and outside the parenthesis.
- Every fact/rule ends with a dot (.).

So, a typical prolog fact consist of following Format:

relation(entity1, entity2,k'th entity).

Example :

- 1.friends(raju, mahesh).
- 2.singer(sonu).
- 3.odd_number(5).

Explanation:

These facts can be interpreted as:

- 1.raju and mahesh are friends.

- 2. sonu is a singer.
- 3.5 is an odd number.

Rules

This term rule is implicit relationship between object and properties of object\

Ex:- 'A' is child of 'B' if 'B' is parent of 'A'

Queries

Queries term related to asking question about relationship between object and object properties

Ex:- 'B' is parent of whom?

Predicates

A predicate is defined by a collection of clauses. A clause is either a rule or a fact. Predicate is formed with the help of facts.

Running queries In Prolog :A typical prolog query can be asked as :

Query 1 : ?- singer(sonu).

Output : Yes.

Explanation : Because our knowledge base contains the above fact, so output was 'Yes', otherwise it would have been 'No'.

Query 2 : ?- odd_number(7).

Output : No.

Explanation : As our knowledge base does not contain the above fact, so output was 'No'.

Sample Tutorials On Prolog :Tutorial:-01(Applying Rules on Facts)

Facts are mentioned below-

- Ram is male
- Shyam is male
- Ravi is male
- Priyanka is female
- Sonia is female

From above facts Predicates are formed as follows

- male(ram).
- male(shyam).
- male(ravi).
- female(priyanka).
- female(sonia).

Program: (Expert_System_Medical_Diagnosys.pl)

go:-

hypothesis(Disease),

write('It is suggested that the patient has '),

```
write(Disease),  
nl,  
undo;  
write('Sorry, the system is unable to identify the disease'),nl,undo.
```

```
hypothesis(cold):-  
symptom(headache),  
symptom(runny_nose),  
symptom(sneezing),  
symptom(sore_throat),  
nl,  
write('Advices and Sugestions:'),  
nl,  
write('1: Tylenol'),  
nl,  
write('2: Panadol'),  
nl,  
write('3: Nasal spray'),  
nl,  
write('Please weare warm cloths because'),  
nl,!.
```

```
hypothesis(influenza) :-  
symptom(sore_throat),  
symptom(fever),  
symptom(headache),  
symptom(chills),  
symptom(body_ache),  
nl,  
write('Advices and Sugestions:'),  
nl,  
write('1: Tamiflu'),  
nl,  
write('2: Panadol'),  
nl,  
write('3: Zanamivir'),  
nl,  
write('Please take a warm bath and do salt gargling because'),  
nl,!.
```

```
hypothesis(typhoid) :-  
symptom(headache),  
symptom(abdominal_pain),  
symptom(poor_appetite),  
symptom(fever),  
nl,  
write('Advices and Sugestions:'),  
nl,
```

```
write('1: Chloramphenicol'),
nl,
write('2: Amoxicillin'),
nl,
write('3: Ciprofloxacin'),
nl,
write('4: Azithromycin'),
nl,
write('Please do complete bed rest and take soft diet because'),
nl,!.

```

```
hypothesis(chicken_pox) :-
symptom(rash),
symptom(body_ache),
symptom(fever),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Varicella vaccine'),
nl,
write('2: Immunoglobulin'),
nl,
write('3: Acetomenaphin'),
nl,
write('4: Acyclovir'),
nl,
write('Please do have oatmeal bath and stay at home because'),
nl.

```

```
hypothesis(measles) :-
symptom(fever),
symptom(runny_nose),
symptom(rash),
symptom(conjunctivitis),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Tylenol'),
nl,
write('2: Aleve'),
nl,
write('3: Advil'),
nl,
write('4: Vitamin A'),
nl,
write('Please get rest and use more liquid because'),
nl,!.

```

```
hypothesis(malaria) :-  
symptom(fever),  
symptom(sweating),  
symptom(headache),  
symptom(nausea),  
symptom(vomiting),  
symptom(diarrhea),  
nl,  
write('Advices and Sugestions:'),  
nl,  
write('1: Aralen'),  
nl,  
write('2: Quaalquin'),  
nl,  
write('3: Plaquenil'),  
nl,  
write('4: Mefloquine'),  
nl,  
write('Please do not sleep in open air and cover your full skin because'),  
nl,!.  
  
ask(Question) :-  
write('Does the patient has the symptom '),  
write(Question),  
write('? : '),  
read(Response),  
nl,  
( (Response == yes ; Response == y)  
->  
assert(yes(Question)) ;  
assert(no(Question)), fail).  
:- dynamic yes/1,no/1.  
  
symptom(S) :-  
(yes(S)  
->  
true ;  
(no(S)  
->  
fail ;  
ask(S))).  
  
undo :- retract(yes()),fail. undo :- retract(no()),fail.  
undo.
```

Program Output:

?- go.

Does the patient has the symptom headache? : y

|: .

Does the patient has the symptom runny_nose? : |: n.

Does the patient has the symptom sore_throat? : |: y.

Does the patient has the symptom fever? : |: y.

Does the patient has the symptom chills? : |: y.

Does the patient has the symptom body_ache? : |: y.

Advices and Sugestions:

1: Tamiflu

2: Panadol

3: Zanamivir

Please take a warm bath and do salt gargling because

It is suggested that the patient has influenza

true.

Facilities: Fedora Operating Systems, Python

Application:

- The following table shows where ES can be applied.

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

Input:

Response to the query asked by the Expert System related to symptoms of the patient.

Output:

The prediction of the disease based on the symptoms provided by patient.

Conclusion:

The Expert System is implemented using PROLOG language which predicts the disease based on the symptoms of the patient.

Questions:

1. What is Expert System?
2. What are the applications of Expert Systems?
3. How expert system is used in Robotics design?
4. Why the languages like PROLOG or LISP are used in Expert System?
5. What is Propositional Logic and how it is used in the design of Expert System?
6. What is First Ordered Logic and how it is used in the design of Expert System?

Experiment No: 6

Title: Develop an elementary chatbot for any suitable customer interaction application.

Aim:

Implement chatbot which interact with the customer and responds to the query asked by the customer.

Prerequisites:

Chatbot implementation concepts, python, chatbot libraries.

Objectives:

Student should be able to customer care chatbot.

Theory:

What is a Chatbot?

A chatbot is an AI-based software designed to interact with humans in their natural languages. These chatbots are usually converse via auditory or textual methods, and they can effortlessly mimic human languages to communicate with human beings in a human-like manner

Chatbots can be categorized into two primary variants – **Rule-Based and Self-learning.**

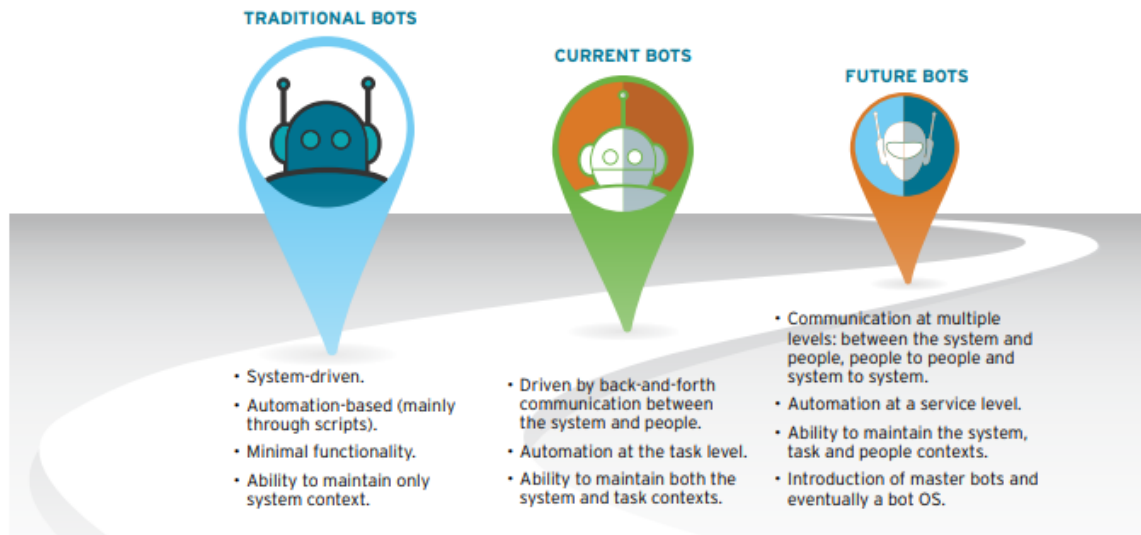
The Rule-based approach trains a chatbot to answer questions based on a set of pre-determined rules on which it was initially trained. These set rules can either be very simple or very complex. While rule-based chatbots can handle simple queries quite well, they usually fail to process more complicated queries/requests.

As the name suggests, self-learning bots are chatbots that can learn on their own. These leverage advanced technologies like Artificial Intelligence and Machine Learning to train themselves from instances and behaviours. Naturally, these chatbots are much smarter than rule-based bots. Self-learning bots can be further divided into two categories – Retrieval Based or Generative.

Chatbot in Today's Generation

Today, we have smart AI-powered Chatbots that use natural language processing (NLP) to understand human commands (text and voice) and learn from experience. Chatbots have become a staple customer interaction tool for companies and brands that have an active online presence (website and social network platforms).

The Bot Evolution



hatterBot Library

ChatterBot is a Python library that is designed to deliver automated responses to user inputs. It makes use of a combination of ML algorithms to generate many different types of responses. This feature allows developers to build chatbots using python that can converse with humans and deliver appropriate and relevant responses. Not just that, the ML algorithms help the bot to improve its performance with experience.

Another excellent feature of ChatterBot is its language independence. The library is designed in a way that makes it possible to train your bot in multiple programming languages.

How does ChatterBot function?

When a user enters a specific input in the chatbot (developed on ChatterBot), the bot saves the input along with the response, for future use. This data (of collected experiences) allows the chatbot to generate automated responses each time a new input is fed into it.

The program chooses the most-fitting response from the closest statement that matches the input, and then delivers a response from the already known selection of statements and responses. Over time, as the chatbot engages in more interactions, the accuracy of response improves.

How To Make A Chatbot In Python?

1. Prepare the Dependencies

The first step in creating a chatbot in Python with the ChatterBot library is to install the library in your system. It is best if you create and use a new Python virtual environment for the installation. To do so, you have to write and execute this command in your Python terminal:

```
pip install chatterbot  
pip install chatterbot_corpus
```

You can also install ChatterBot's latest development version directly from GitHub. For this, you will have to write and execute the following command:

```
pip install git+git://github.com/gunthercox/ChatterBot.git@master
```

If you wish to upgrade the command, you can do so as well:

```
pip install --upgrade chatterbot_corpus  
pip install --upgrade chatterbot
```

Now that your setup is ready, we can move on to the next step to create chatbot using python.

2. Import Classes

Importing classes is the second step in the Python chatbot creation process. All you need to do is import two classes – ChatBot from chatterbot and ListTrainer from chatterbot.trainers. To do this, you can execute the following command:

```
from chatterbot import ChatBot  
from chatterbot.trainers import ListTrainer
```

3. Create and Train the Chatbot

This is the third step on creating chatbot in python. The chatbot you are creating will be an instance of the class “ChatBot.” After creating a new ChatterBot instance, you can train the bot to improve its performance. Training ensures that the bot has enough knowledge to get started with specific responses to specific inputs. You have to execute the following command now:

```
my_bot = ChatBot(name='PyBot', read_only=True,
                  logic_adapters=
                  ['chatterbot.logic.MathematicalEvaluation',
                   'chatterbot.logic.BestMatch'])
```

Here, the argument (that corresponds to the parameter name) represents the name of your Python chatbot. If you wish to disable the bot's ability to learn after the training, you can include the “read_only=True” command. The command “logic_adapters” denotes the list of adapters used to train the chatbot.

While the “chatterbot.logic.MathematicalEvaluation” helps the bot to solve math problems, the “chatterbot.logic.BestMatch” helps it to choose the best match from the list of responses already provided.

Since you have to provide a list of responses, you can do it by specifying the lists of strings that can be later used to train your Python chatbot, and find the best match for each query. Here's an example of responses you can train your chatbot using python to learn:

```
small_talk = ['hi there!',
              'hi!',
              'how do you do?',
              'how are you?',
              'i\'m cool.',
              'fine, you?',
              'always cool.',
              'i\'m ok',
              'glad to hear that.',
              'i\'m fine',
              'glad to hear that.',
              'i feel awesome',
              'excellent, glad to hear that.',
              'not so good',
              'sorry to hear that.',
              'what\'s your name?',
              'i\'m pybot. ask me a math question, please.'],

math_talk_1 = ['pythagorean theorem',
               'a squared plus b squared equals c squared.'],

math_talk_2 = ['law of cosines',
               'c**2 = a**2 + b**2 - 2 * a * b * cos(gamma)']
```

You can also create and train the bot by writing an instance of “ListTrainer” and supplying it with a list of strings like so:

```
list_trainer = ListTrainer(my_bot)

for item in (small_talk, math_talk_1, math_talk_2):
    list_trainer.train(item)
```

Now, your Python chatbot is ready to communicate.

4. Communicate with the Python Chatbot

To interact with your Python chatbot, you can use the `.get_response()` function. This is how it should look while communicating:

```
>>> print(my_bot.get_response("hi"))
how do you do?

>>> print(my_bot.get_response("i feel awesome today"))
excellent, glad to hear that.

>>> print(my_bot.get_response("what's your name?"))
i'm pybot. ask me a math question, please.

>>> print(my_bot.get_response("show me the pythagorean
theorem"))
a squared plus b squared equals c squared.

>>> print(my_bot.get_response("do you know the law of
cosines?"))

$$c^2 = a^2 + b^2 - 2 * a * b * \cos(\gamma)$$

```

However, it is essential to understand that the chatbot using python might not know how to answer all your questions. Since its knowledge and training is still very limited, you have to give it time and provide more training data to train it further.

5. Train your Python Chatbot with a Corpus of Data

In this last step of how to make a chatbot in Python, for training your python chatbot even further, you can use an existing corpus of data. Here's an example of how to train your Python chatbot with a corpus of data provided by the bot itself:

```
from chatterbot.trainers import ChatterBotCorpusTrainer

corpus_trainer = ChatterBotCorpusTrainer(my_bot)
corpus_trainer.train('chatterbot.corpus.english')
```

Code:

```
# Importing chatterbot
```

```
from chatterbot import ChatBot
```

```
# Create object of ChatBot class
```

```
bot = ChatBot('Buddy')
```

```
# Create object of ChatBot class with Storage Adapter
```

```
bot = ChatBot(
    'Buddy',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    database_uri='sqlite:///database.sqlite3'
)
```

```
# Create object of ChatBot class with Logic Adapter
```

```
bot = ChatBot(
    'Buddy',
    logic_adapters=[
        'chatterbot.logic.BestMatch',
        'chatterbot.logic.TimeLogicAdapter'],
)
```

```
# Inport ListTrainer

from chatterbot.trainers import ListTrainer

trainer = ListTrainer(bot)

trainer.train([

    'Hi',

    'Hello',

    'I need your assistance regarding my order',

    'Please, Provide me with your order id',

    'I have a complaint.',

    'Please elaborate, your concern',

    'How long it will take to receive an order ?',

    'An order takes 3-5 Business days to get delivered.',

    'Okay Thanks',

    'No Problem! Have a Good Day!'

])

# Get a response to the input text 'I would like to book a flight.'

response = bot.get_response('I have a complaint.')

print("Bot Response:", response)
```

```
name=input("Enter Your Name: ")  
  
print("Welcome to the Bot Service! Let me know how can I help you?")  
  
while True:  
  
    request=input(name+':')  
  
    if request=='Bye' or request == 'bye':  
  
        print('Bot: Bye')  
  
        break  
  
    else:  
  
        response=bot.get_response(request)  
  
        print('Bot:',response)
```

Assignment No: 7

Aim: Case study on Amazon EC2 to learn about Amazon EC2, Amazon Elastic Compute Cloud is a central

part of Amazon.com's cloud computing platform, Amazon Web Services. How EC2 allows users to rent virtual computers on which to run their own computer applications.

Objectives:

1. To learn Amazon Web Services.
2. To case study the Amazon EC2.

Software Requirements:

Ubuntu 18.04 PHP
MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Amazon Elastic Compute Cloud (EC2)

Elastic IP addresses allow you to allocate a static IP address and programmatically assign it to an instance. You can enable monitoring on an Amazon EC2 instance using Amazon CloudWatch² in order to gain visibility into resource utilization, operational performance and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic). You can create Auto Scaling³ to automatically scale your capacity on certain conditions based on metrics that Amazon CloudWatch collects. You can also distribute incoming traffic by creating an elastic load balancer using the Elastic Load Balancing⁴ service. Amazon Elastic Block Storage (EBS)⁵ volumes provide network-attached, high-performance storage for Amazon EC2 instances. Point-in-time consistent snapshots of EBS volumes can be created and stored on Amazon Simple Storage Service (Amazon S3)⁶. Amazon S3 is a highly durable and distributed data store. With a simple web services interface, you can store and retrieve data at any time, from anywhere on the web using standard HTTP verbs. Copies of objects can be distributed and cached at 14 edge locations around the world by creating a distribution using Amazon CloudFront⁷ service (content). Amazon SimpleDB⁸ is a web service that provides the core functionality of a database- real-time lookup and simple querying of structured data

complexity. You can organize the dataset into domains and can run queries across all of the Auto-scaling Group using the Auto network-attached persistent storage to Amazon EC2 instances. retrieve large amounts of data as objects in buckets (containers) udFront7 – a web service for content delivery (static or streaming) – without the operational plexity. performance,Auto-scaling metric that n even data stored in a particular domain. Domains are collections of items that are described by attribute-value pairs.

Amazon Relational Database Service⁹ (Amazon RDS) provides an easy way to setup, operate and scale a relational database in the cloud. You can launch a DB Instance and get access to a full-featured MySQL database and not worry about common database administration tasks like backups, patch management etc.

Amazon Simple Queue Service (Amazon SQS)¹⁰ is a reliable, highly scalable, hosted distributed queue for storing messages as they travel between computers and application components.

Amazon Simple Notifications Service (Amazon SNS) provides a simple way to notify applications or people from the cloud by creating Topics and using a publish-subscribe protocol.

Amazon Elastic MapReduce provides a hosted Hadoop framework running on the webscale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) and allows you to create customized JobFlows. JobFlow is a sequence of MapReduce steps.

Amazon Virtual Private Cloud (Amazon VPC) allows you to extend your corporate network into a private cloud contained within AWS. Amazon VPC uses IPsec tunnel mode that enables you to create a secure connection between a gateway in your data center and a gateway in AWS.

Amazon Route⁵³ is a highly scalable DNS service that allows you manage your DNS records by creating a HostedZone for every domain you would like to manage.

AWS Identity and Access Management (IAM) enable you to create multiple Users with unique security credentials and manage the permissions for each of these Users within your AWS Account. IAM is natively integrated into AWS Services. No service APIs have changed to support IAM, and exiting applications and tools built on top of the AWS service APIs will continue to work when using IAM. AWS also offers various payment and billing services that leverages Amazon's payment infrastructure. All AWS infrastructure services offer utility-style pricing that require no longterm commitments or contracts. For example, you pay by the hour for Amazon EC2 instance usage and pay by the gigabyte for storage and data transfer in the case of Amazon S3. More information about each of these services and their pay-as-you-go pricing is available on the

You are free to use the programming model, language, or operating system (Windows, OpenSolaris or any flavor of Linux) of your choice.

You are free to pick and choose the AWS products that best satisfy your requirements—you can use any of the services individually or in any combination.

Because AWS provides resizable (storage, bandwidth and computing) resources, you are free to consume as much or as little and only pay for what you consume.

You are free to use the system management tools you've used in the past and extend your datacenter into the cloud.

Launching an EC2 instance

1. Sign in to the preview version of the [AWS Management Console](#)
2. Open the Amazon EC2 console by choosing **EC2** under Compute.

3. From the EC2 Console, click **Launch Instance**.

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	1	Dedicated Hosts	0	Elastic IPs	5
Instances (all states)	13	Key pairs	1	Load balancers	2
Placement groups	1	Security groups	51	Snapshots	19
Volumes	18				

Launch instance

To get started, launch an Amazon EC2 Instance, which is a virtual server in the cloud.

Launch Instance

Note: Your instances will launch in the US East (N. Virginia) Region.

Service health

Region: US East (N. Virginia) Status: ✔ This service is operating normally

Zone status

The **Choose an Amazon Machine Image (AMI)** page displays a list of basic configurations called Amazon Machine Images (AMIs) that serve as templates for your instance. Select the HVM edition of the **Amazon Linux 2 AMI**.

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application servers, and applications) required to launch your instances. You can select an AMI provided by AWS, not available publicly, or you can select one of your own AMIs.

Search for an AMI by entering a keyword, such as "Ubuntu".

Quick Start

My AMIs

AWS Marketplace

Community AMIs

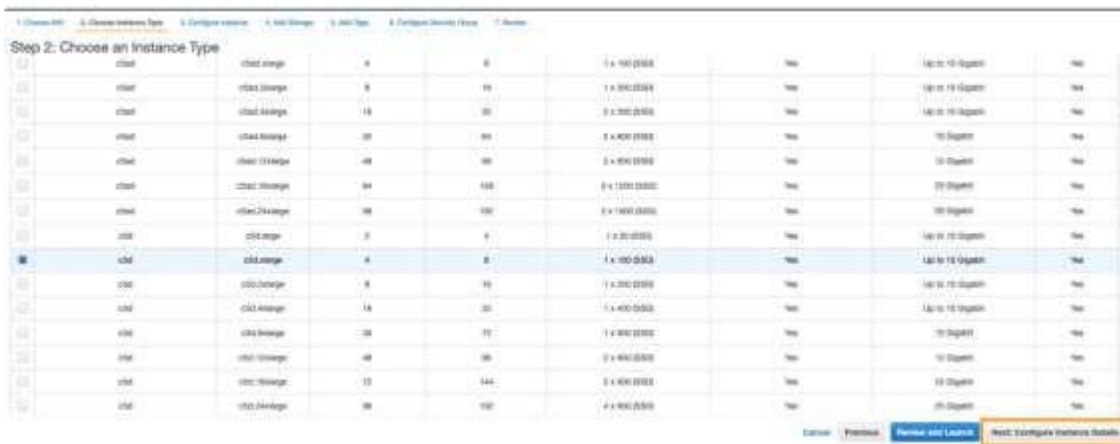
Amazon Linux 2 AMI (HVM, BIOS) Volume Type - ami-0a2209996a071237 (64-bit x86) / ami-0310084f70210a0d (64-bit ARM)

Amazon Linux 2 comes with free-year support. It provides Linux kernel 4.14 based on optimal performance for Amazon EC2, supports x86_64, ARM, and AWS Graviton 2. It is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020, and has been deprecated since this point.

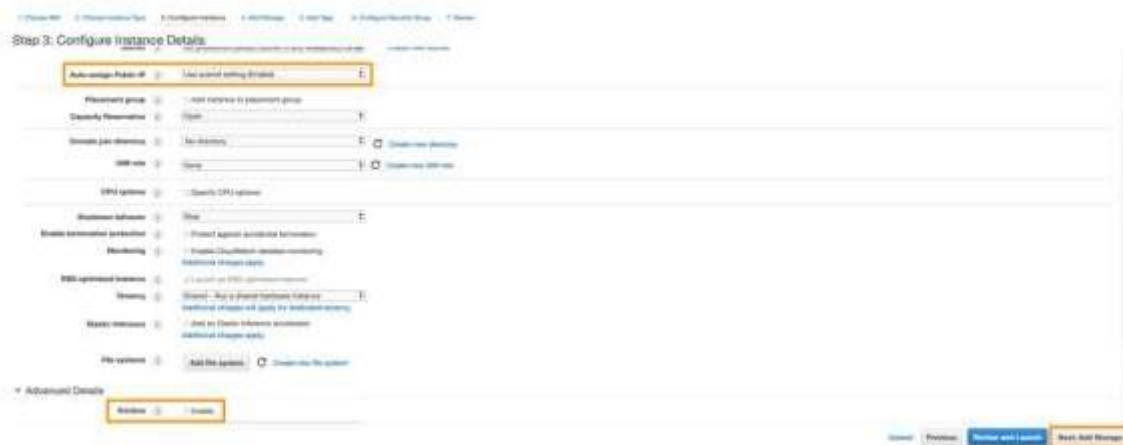
See developer AMIs | View Amazon Linux AMIs | See Amazon Linux 2 AMIs

Launch

On the **Choose an Instance Type** page, choose **c5d.xlarge** as the hardware configuration of your instance and **Review and Launch**.



On Instances details, make sure the Auto-assign Public IP is **Enable** and you selected Enclave as **Enable**. Click on **Next: Add Storage**



Review the configurations and click **next: Add Tapes**The **ephemeral0** is the Instance Storage based on NVMe SSD.

Step 4: Add Storage

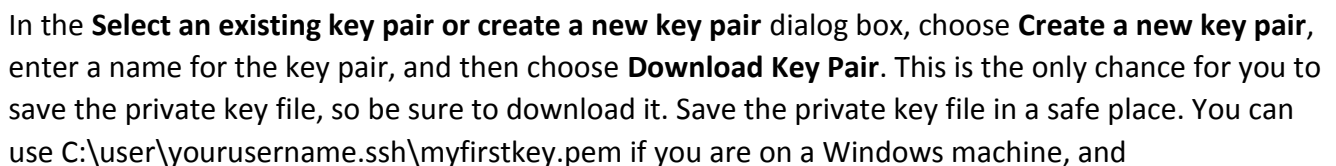
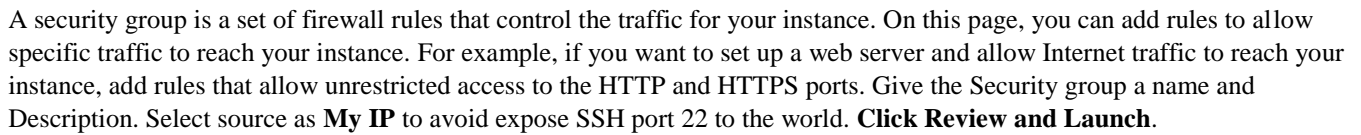
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snaps-0740be58d4d1f11a	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
ephemeral0	/dev/nvme0n1	N/A	100	NVMe SSD	N/A	N/A	N/A	Hardware Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Add a tag and click **Next: Configure Security Group**



~/ssh/myfirstkey.pem if you are on a Mac or Linux machine. You need to provide the name of your key pair when you launch an instance, and the corresponding private key each time you connect to the instance.

Select an existing key pair or create a new key pair ×

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name

MyKeypair

Download Key Pair

...

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

A confirmation page lets you know that your instance is launching. Choose **View Instances** to close the confirmation page and return to the console.

On the Instances page, you can view the status of your instance. It takes a short time for an instance to launch. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running, and it receives a public DNS name.

Conclusion

Performed case study of Amazon web services: Amazon EC2.

Assignment No: 7

Aim: Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. How it work, different services provided by it.

Objectives:

1. To learn Microsoft Azure Cloud computing platform.
2. To case study the Microsoft Azure cloud services.

Software Requirements:

Ubuntu 18.04 PHP
MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Execution Environment

The Windows Azure execution environment consists of a platform for applications and services hosted within one or more roles. The types of roles you can implement in Windows Azure are:

- **Azure Compute (Web and Worker Roles).** A Windows Azure application consists of one or more hosted roles running within the Azure data centers. Typically there will be at least one Web role that is exposed for access by users of the application. The application may contain additional roles, including Worker roles that are typically used to perform background processing and support tasks for Web roles. For more detailed information see “Overview of Creating a Hosted Service for Windows Azure” at <http://technet.microsoft.com/en-au/library/gg432976.aspx> and “Building an Application that Runs in a Hosted Service” at <http://technet.microsoft.com/en-au/library/hh180152.aspx>.
- **Virtual Machine (VM role).** This role allows you to host your own custom instance of the Windows Server 2008 R2 Enterprise or Windows Server 2008 R2 Standard operating system within a Windows Azure data center. For more detailed information see “Creating Applications by Using a VM Role in Windows Azure” at <http://technet.microsoft.com/enau/>

library/gg465398.aspx.

Data Management

Windows Azure, SQL Azure, and the associated services provide opportunities for storing and managing data in a range of ways. The following data management services and features are available:

- **Azure Storage:** This provides four core services for persistent and durable data storage in the cloud. The services support a REST interface that can be accessed from within Azurehosted or on-premises (remote) applications. For information about the REST API, see “Windows Azure Storage Services REST API Reference” at <http://msdn.microsoft.com/enus/library/dd179355.aspx>. The four storage services are:

- **The Azure Table Service** provides a table-structured storage mechanism based on the familiar rows and columns format, and supports queries for managing the data. It is primarily aimed at scenarios where large volumes of data must be stored, while being easy to access and update. For more detailed information see “Table Service Concepts” at <http://msdn.microsoft.com/en-us/library/dd179463.aspx> and “Table Service API” at <http://msdn.microsoft.com/en-us/library/dd179423.aspx>.

- **The Binary Large Object (BLOB) Service** provides a series of containers aimed at storing text or binary data. It provides both Block BLOB containers for streaming data, and Page BLOB containers for random read/write operations. For more detailed information see “Understanding Block Blobs and Page Blobs” at <http://msdn.microsoft.com/en-us/library/ee691964.aspx> and “Blob Service API” at <http://msdn.microsoft.com/en-us/library/dd135733.aspx>.

- **The Queue Service** provides a mechanism for reliable, persistent messaging between role instances, such as between a Web role and a Worker role. For more detailed information see “Queue Service Concepts” at <http://msdn.microsoft.com/enus/library/dd179353.aspx> and “Queue Service API” at <http://msdn.microsoft.com/enus/library/dd179363.aspx>.

- Windows Azure Drives provide a mechanism for applications to mount a single volume NTFS VHD as a Page BLOB, and upload and download VHDs via the BLOB. For more detailed information see “Windows Azure Drive” (PDF) at <http://go.microsoft.com/?linkid=9710117>.

- **SQL Azure Database:** This is a highly available and scalable cloud database service built on SQL Server technologies, and supports the familiar T-SQL based relational database model. It can be used with applications hosted in Windows Azure, and with other applications running on-premises or hosted elsewhere. For more detailed information see “SQL Azure Database” at <http://msdn.microsoft.com/en->

[us/library/ee336279.aspx](http://msdn.microsoft.com/en-us/library/ee336279.aspx).

- **Data Synchronization:** SQL Azure Data Sync is a cloud-based data synchronization service built on Microsoft Sync Framework technologies. It provides bi-directional data synchronization and data management capabilities allowing data to be easily shared between multiple SQL Azure databases and between on-premises and SQL Azure databases.

For

more detailed information see “Microsoft Sync Framework Developer Center” at <http://msdn.microsoft.com/en-us/sync>.

- **Caching:** This service provides a distributed, in-memory, low latency and high throughput application cache service that requires no installation or management, and dynamically increases and decreases the cache size automatically as required. It can be used to cache application data, ASP.NET session state information, and for ASP.NET page output caching. For more detailed information see “Caching Service (Windows Azure AppFabric)” at <http://msdn.microsoft.com/en-us/library/gg278356.aspx>.

Networking Services

Windows Azure provides several networking services that you can take advantage of to maximize performance, implement authentication, and improve manageability of your hosted applications. These services include the following:

- **Content Delivery Network (CDN).** The CDN allows you to cache publicly available static data for applications at strategic locations that are closer (in network delivery terms) to end users. The CDN uses a number of data centers at many locations around the world, which store the data in BLOB storage that has anonymous access. These do not need to be locations where the application is actually running. For more detailed information see “Delivering High-Bandwidth Content with the Windows Azure CDN” at <http://msdn.microsoft.com/en-us/library/ee795176.aspx>.
- **Virtual Network Connect.** This service allows you to configure roles of an application running in Windows Azure and computers on your on-premises network so that they appear to be on the same network. It uses a software agent running on the on-premises computer to establish an IPsec-protected connection to the Windows Azure roles in the cloud, and provides the capability to administer, manage, monitor, and debug the roles directly. For more detailed information see “Connecting Local Computers to Windows Azure Roles” at <http://msdn.microsoft.com/en-us/library/gg433122.aspx>.
- **Virtual Network Traffic Manager.** This is a service that allows you to set up request redirection and load balancing based on three different methods. Typically you will use Traffic Manager to maximize performance by redirecting requests from users to the instance in the closest data center using the Performance method. Alternative load balancing methods available are Failover and Round Robin. For more detailed information see “Windows Azure Traffic Manager” at http://msdn.microsoft.com/en-us/WAZPlatformTrainingCourse_WindowsAzureTrafficManager.
- **Access Control.** This is a standards-based service for identity and access control that makes use of a range of identity providers (IdPs) that can authenticate users. ACS acts as a Security Token Service (STS), or token issuer, and makes it easier to take advantage of federation authentication techniques where user identity is validated in a realm or domain other than that in which the application resides. An example is controlling user access based on an identity verified by an identity provider such as Windows Live ID or Google. For more detailed information see “Access Control Service 2.0” at <http://msdn.microsoft.com/en-us/library/gg429786.aspx> and “Claims Based Identity & Access Control Guide” at <http://claimsid.codeplex.com/>.

- **Service Bus.** This provides a secure messaging and data flow capability for distributed and hybrid applications, such as communication between Windows Azure hosted applications and on-premises applications and services, without requiring complex firewall and security infrastructures. It can use a range of communication and messaging protocols and patterns to provide delivery assurance, reliable messaging; can scale to accommodate varying loads; and can be integrated with on-premises BizTalk Server artifacts. For more detailed information see “AppFabric Service Bus” at <http://msdn.microsoft.com/en-us/library/ee732537.aspx>

Conclusion

Performed case study of Microsoft Azure Cloud computing platform and services.

Assignment No: 8

Aim: Assignment to install and configure Google App Engine.

Objectives:

1. To learn basics of Google App Engine.
2. To install and configure Google App Engine.

Software Requirements:

Ubuntu 18.04
Python MySQL

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

Google App Engine is Google's platform as a service offering that allows developers and businesses to build and run applications using Google's advanced infrastructure. These applications are required to be written in one of a few supported languages, namely: Java, Python, PHP and Go. It also requires the use of Google query language and that the database used is Google Big Table.

Applications must abide by these standards, so applications either must be developed with GAE in mind or else modified to meet the requirements. GAE is a platform, so it provides all of the required elements to run and host Web applications, be it on mobile or Web. Without this all-in feature, developers would have to source their own servers, database software and the APIs that would make all of them work properly together, not to mention the entire configuration that must be done. GAE takes this burden off the developers so they can concentrate on the app front end and functionality, driving better user experience.

Advantages of GAE include:

- Readily available servers with no configuration requirement
- Power scaling function all the way down to "free" when resource usage is minimal
- Automated cloud computing tools

1. Make sure you have python installed in your ubuntu system. run the command `"python - V"` and most probably you will get "Python 2.7.6" or above.
2. Crul <https://sdk.cloud.google.com> and use bash to run the commands by typing this command `curl https://sdk.cloud.google.com | bash`
3. Whenever you get to choose directories just hit enter, "YEAH IT WILL BE FINE".
4. Follow the instructions in the installation process.
5. Then run `gcloud init`
6. Follow the installation instructions as they are very straight forward.
7. Choose the account you want to use for google app engine.
8. Choose the project with numeric choice (don't use textual, you might make mistake). If you do not already have a google app engine project create a app engine project by following this link. <https://console.cloud.google.com/start>
9. Enable google api by pressing Y in the command line prompt.

Now as we have finished installing appengine, now it's time to create and upload an app. In this case we will be taking example of a "HELLO WORLD" app in python.

1. As we already have made sure that we have python installed in our system, It will be easier for us to clone existing code and deploy it rather than creating our own so we will use python-docs-sample. Run the command `"git clone https://github.com/GoogleCloudPlatform/python-docs-samples"`.
2. cd to hello world sample by typing the command `" cd python-docssamples/ appengine/standard/hello_world"`.
3. Then run the command `"dev_appserver.py app.yml"`. It will run and give you the url of default and admin. If you go to the link of default you see the text hello world like this.

This is how you run the python app in your local server. But what we have to do is hosting the app in google app engine. To do so Now let's follow the following instructions.

1. Run the command `Ctrl + C`.
2. Being in the same working directory hello-world run the command `gcloud app deploy`
3. Select the project you want to deploy the app , press Y and enter to continue. after that you will get the console output `"Deployed service[default] to [Your web url for appengine]"`
4. If you copy and paste the url, you will see the hello world in the browser too. Web output

Now you have successfully uploaded your web app into app engine. Conclusion

Hence we learnt to install and configure Google App Engine.

Assignment No: 9**Aim: Creating an Application in SalesForce.com using Apex programmingLanguage****Objectives:**

3. To learn Sales Force Cloud
4. To study Apex programmingLanguage

Software Requirements:

Ubuntu
18.04 PHP
MySQL
Apex

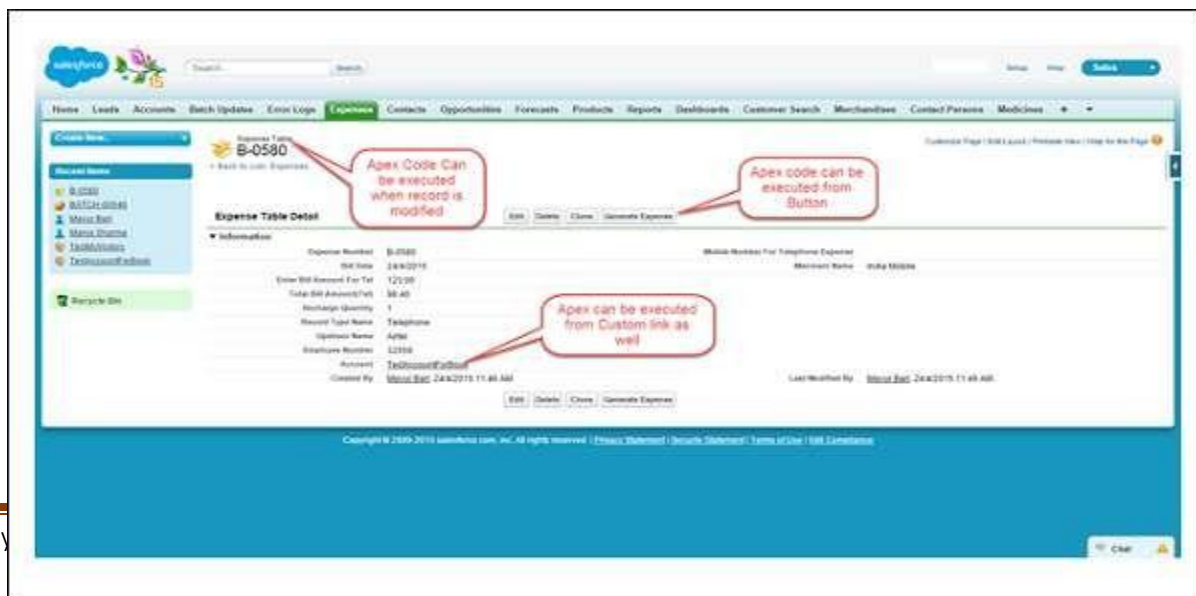
Hardware Requirements:

Pentium IV system with latest configuration

Theory:**What is Apex?**

Apex is a proprietary language developed by the Salesforce.com. As per the official definition, Apex is a strongly typed, object-oriented programming language that allows developers to execute the flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API.

It has a Java-like syntax and acts like database stored procedures. It enables the developers to add business logic to most system events, including button clicks, related record updates, and Visual force **pages**. **Apex** code can be initiated by Web service requests and from triggers on objects. Apex is included in Performance Edition, Unlimited



Edition, Enterprise Edition, and Developer Edition.

Features of Apex as a Language

Let us now discuss the features of Apex as a Language –

➤ **Integrated**

Apex has built in support for DML operations like INSERT, UPDATE, DELETE and also DML Exception handling. It has support for inline SOQL and SOSL query handling which returns the set of sObject records. We will study the sObject, SOQL, SOSL in detail in future chapters.

➤ **Java like syntax and easy to use**

Apex is easy to use as it uses the syntax like Java. For example, variable declaration, loop syntax and conditional statements.

➤ **Strongly Integrated With Data**

Apex is data focused and designed to execute multiple queries and DML statements together. It issues multiple transaction statements on Database.

➤ **Strongly Typed**

Apex is a strongly typed language. It uses direct reference to schema objects like sObject and any invalid reference quickly fails if it is deleted or if it is of wrong data type.

➤ **Multitenant Environment**

Apex runs in a multitenant environment. Consequently, the Apex runtime engine is designed to guard closely against runaway code, preventing it from monopolizing shared resources. Any code that violates limits fails with easy-to-understand error messages.

➤ **Upgrades Automatically**

Apex is upgraded as part of Salesforce releases. We don't have to upgrade it manually.

➤ **Easy Testing**

Apex provides built-in support for unit test creation and execution, including test results that indicate how much code is covered, and which parts of your code can be more efficient.

When Should Developer Choose Apex?

Apex should be used when we are not able to implement the complex business functionality using the pre-built and existing out of the box functionalities. Below are the cases where we need to use apex over Salesforce configuration.

Apex Applications

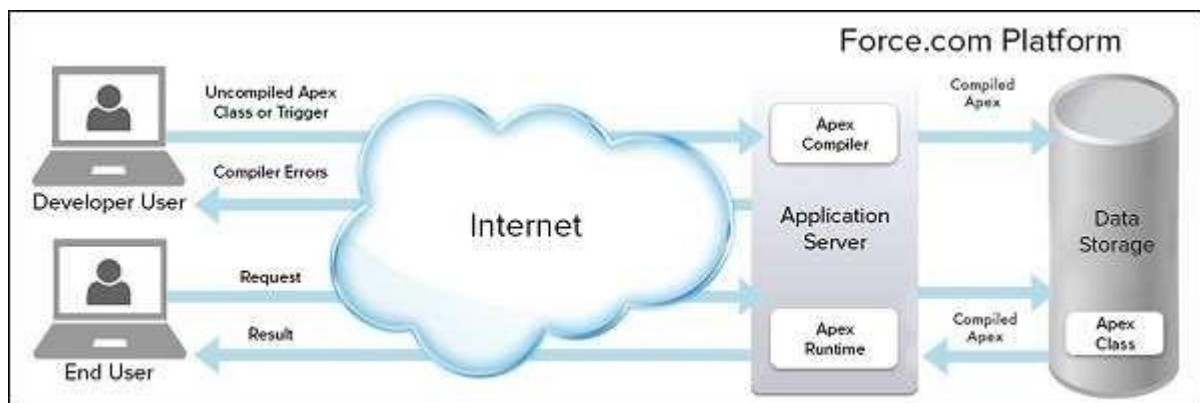
We can use Apex when we want to –

- ❑ Create Web services with integrating other systems.
- ❑ Create email services for email blast or email setup.

- ❑ Perform complex validation over multiple objects at the same time and also customvalidation implementation.
- ❑ Create complex business processes that are not supported by existing workflow functionality or flows.
- ❑ Create custom transactional logic (logic that occurs over the entire transaction, not justwith a single record or object) like using the Database methods for updating the records.
- ❑ Perform some logic when a record is modified or modify the related object's record when there is some event which has caused the trigger to fire.

Working Structure of Apex

As shown in the diagram below (Reference: Salesforce Developer Documentation), Apex runsentirely on demand Force.com Platform



Flow of Actions

There are two sequence of actions when the developer saves the code and when an end user performs some action which invokes the Apex code as shown below –

Developer Action

When a developer writes and saves Apex code to the platform, the platform application server first compiles the code into a set of instructions that can be understood by the Apex runtime interpreter, and then saves those instructions as metadata.

End User Action

When an end-user triggers the execution of Apex, by clicking a button or accessing a Visualforce page, the platform application server retrieves the compiled instructions from the metadata and sends them through the runtime interpreter before returning the result. The end-user observes no differences in execution time as compared to the standard application platform request.

Since Apex is the proprietary language of Salesforce.com, it does not support some features which a general programming language does. Following are a few features which Apex doesnot support –

- ❓ It cannot show the elements in User Interface.
- ❓ You cannot change the standard SFDC provided functionality and also it is not possible to prevent the standard functionality execution.
- ❓ You cannot change the standard SFDC provided functionality and also it is not possible to prevent the standard functionality execution.
- ❓ Creating multiple threads is also not possible as we can do it in other languages.

Understanding the Apex Syntax

Apex code typically contains many things that we might be familiar with from other programming languages.

Variable Declaration

As a strongly typed language, you must declare every variable with data type in Apex. As seen in the code below (screenshot below), `lstAcc` is declared with data type as List of Accounts.

SOQL Query

This will be used to fetch the data from Salesforce database. The query shown in screenshot below is fetching data from Account object.

Loop Statement

This loop statement is used for iterating over a list or iterating over a piece of code for a specified number of times. In the code shown in the screenshot below, iteration will be same as the number of records we have.

Flow Control Statement

The If statement is used for flow control in this code. Based on certain condition, it is decided whether to go for execution or to stop the execution of the particular piece of code. For example, in the code shown below, it is checking whether the list is empty or it contains records.

DML Statement

Performs the records insert, update, upsert, delete operation on the records in database. For example, the code given below helps in updating Accounts with new field value.

Apex Code Development Tools

In all the editions, we can use any of the following three tools to develop the code –

- ❓ Force.com Developer Console
- ❓ Force.com IDE
- ❓ Code Editor in the Salesforce User Interface

Creating an Application in SalesForce.com using Apex programming Language

CreateAccount.apxc

```
public class CreateAccount {  
  
    public String name {get; set;}  
  
    public String phoneNumber {get; set;}  
  
    public String selectedname {get; set;}  
  
    public String websiteURL {get; set;}  
  
    public List<Selectoption> lstnamesel {get; set;}  
  
    public CreateAccount(ApexPages.StandardController controller) {  
  
        lstnamesel = new List<selectoption>();  
  
        lstnamesel.add(new selectOption('','- None -'));  
  
        lstnamesel.add(new selectOption('IT','IT'));  
  
        lstnamesel.add(new selectOption('MECH','MECH'));  
  
        lstnamesel.add(new selectOption('CHEM','CHEM'));  
  
        lstnamesel.add(new selectOption('PHARMA','PHARMA'));  
  
    }  
  
    public PageReference createAccount() {  
  
        System.debug('teset create');  
  
        if(!String.isEmpty(name)) {  
  
            Account accountRecord = new Account(Name = name,  
            Phone = phoneNumber,  
            Industry = selectedname,  
            Website = websiteURL);  
  
            INSERT accountRecord;  
  
            PageReference pg = new PageReference('/'+accountRecord.Id);  
  
            pg.setRedirect(true);  
  
            return pg;  
  
        } else {
```

```

ApexPages.addmessage(new ApexPages.message(ApexPages.severity.CONFIRM,'Please enter
Account Name'));
}
return NULL;
}

public PageReference cancelAccount() {
PageReference pg = new PageReference('/'+Schema.SObjectType.Account.getKeyPrefix()+'/o');
return pg;
}
}

```

CreateAccount.vfp

```

<apex:page standardController="Account" extensions="CreateAccount">
<apex:form id="apexFrom" >
<apex:pageBlock title="Create Account:" id="pageBlockId">
<apex:pageMessages id="showmsg"></apex:pageMessages>
<apex:pageBlockSection columns="2" >
<!--<div class = "requiredInput">
<div class = "requiredBlock"></div>
<apex:inputText value="{!name}" label="Account Name" required="true"/>
</div-->
<apex:inputText value="{!name}" label="Account Name" required="true"/>
<apex:inputText value="{!phoneNumber}" label="Phone" />
<apex:selectList size="1" value="{!selectedname}" label="Industry">
<apex:selectOptions value="{!lstnamesel}"/>
</apex:selectList>
<apex:inputText value="{!websiteURL}" label="Website" />

```

```
</apex:pageBlockSection>

<apex:pageBlockButtons >

<apex:commandButton action="{!createAccount}" value="Save"/>

<apex:commandButton action="{!cancelAccount}" value="Cancel"/>

</apex:pageBlockButtons>

</apex:pageBlock>

<!-- ACTION FUNCTIONS-->

<apex:actionFunction name="createFunction" action="{!createAccount}"/>

<apex:actionFunction name="cancelFunction" action="{!cancelAccount}"/>

</apex:form>

</apex:page>
```

Step No 1:

Create new org:

<https://developer.salesforce.com/signup>

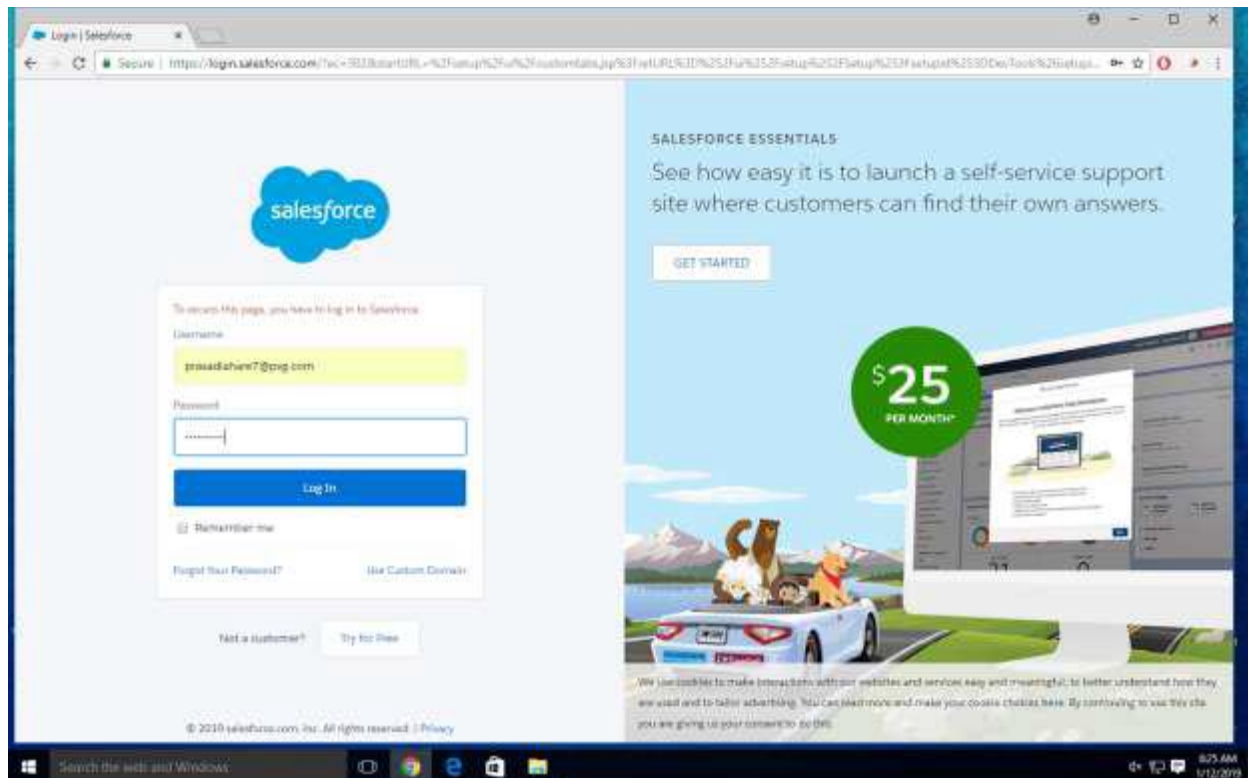
Step No 2:

After signup, logging using following URI

<https://login.salesforce.com/>

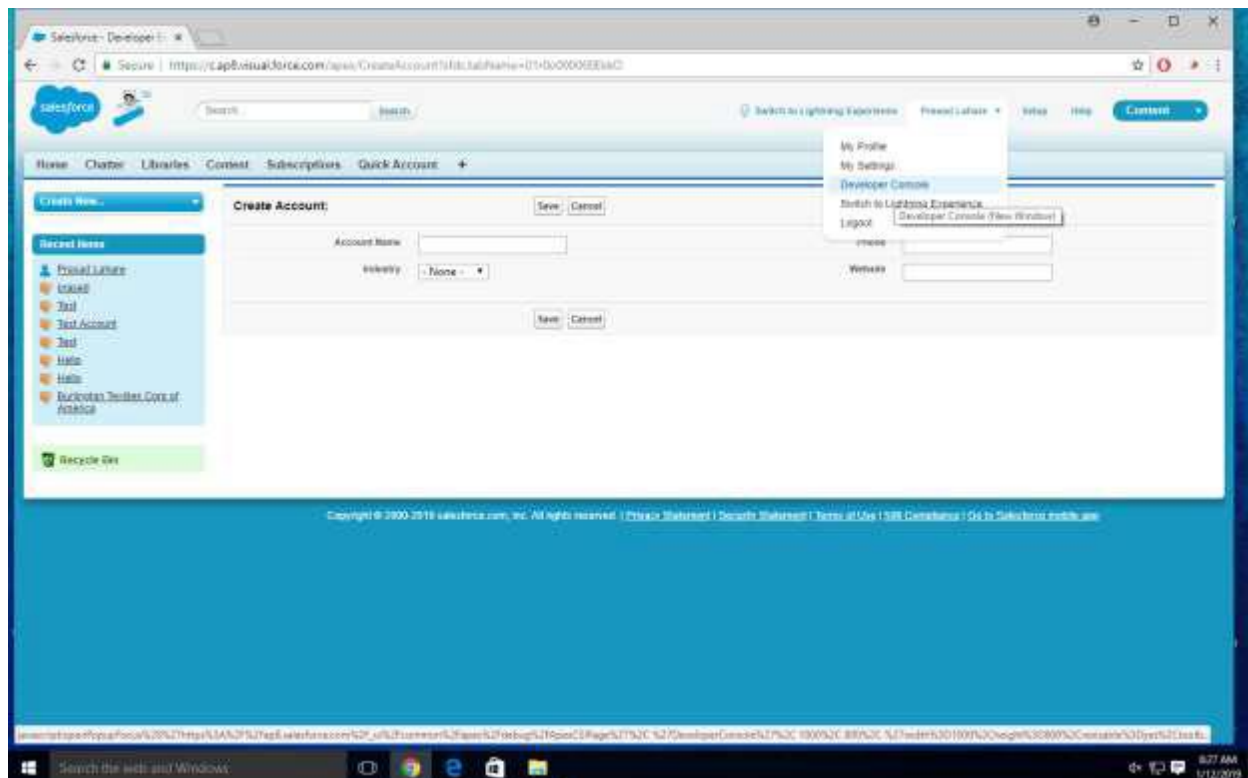
Step No 3:

Login Page



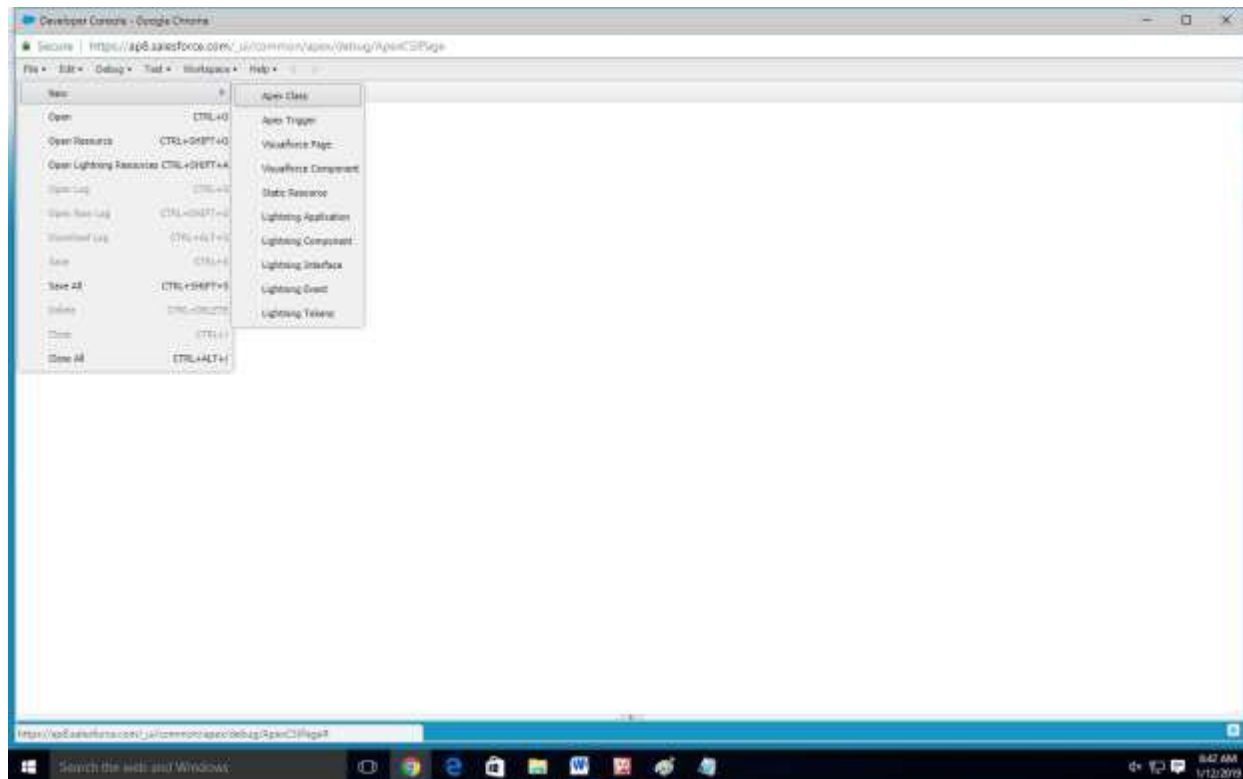
Step No:4

Go to Developer Console for writing Program

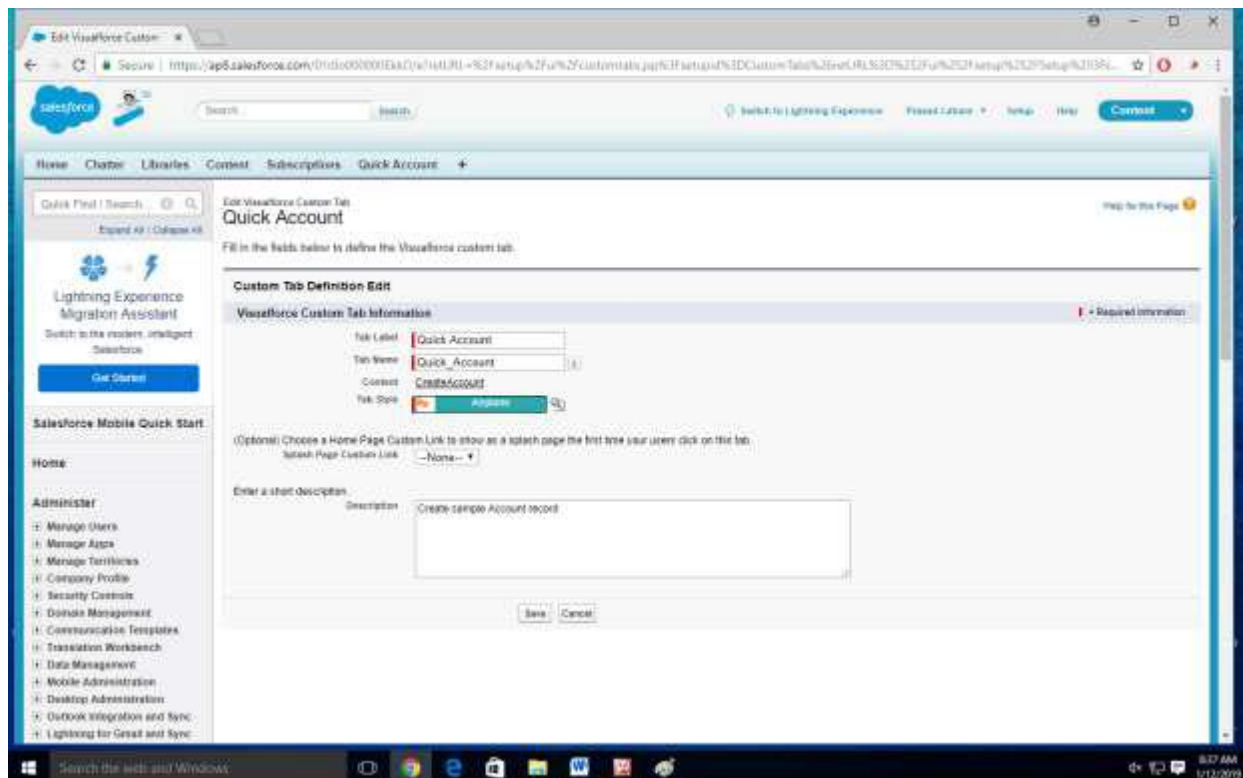
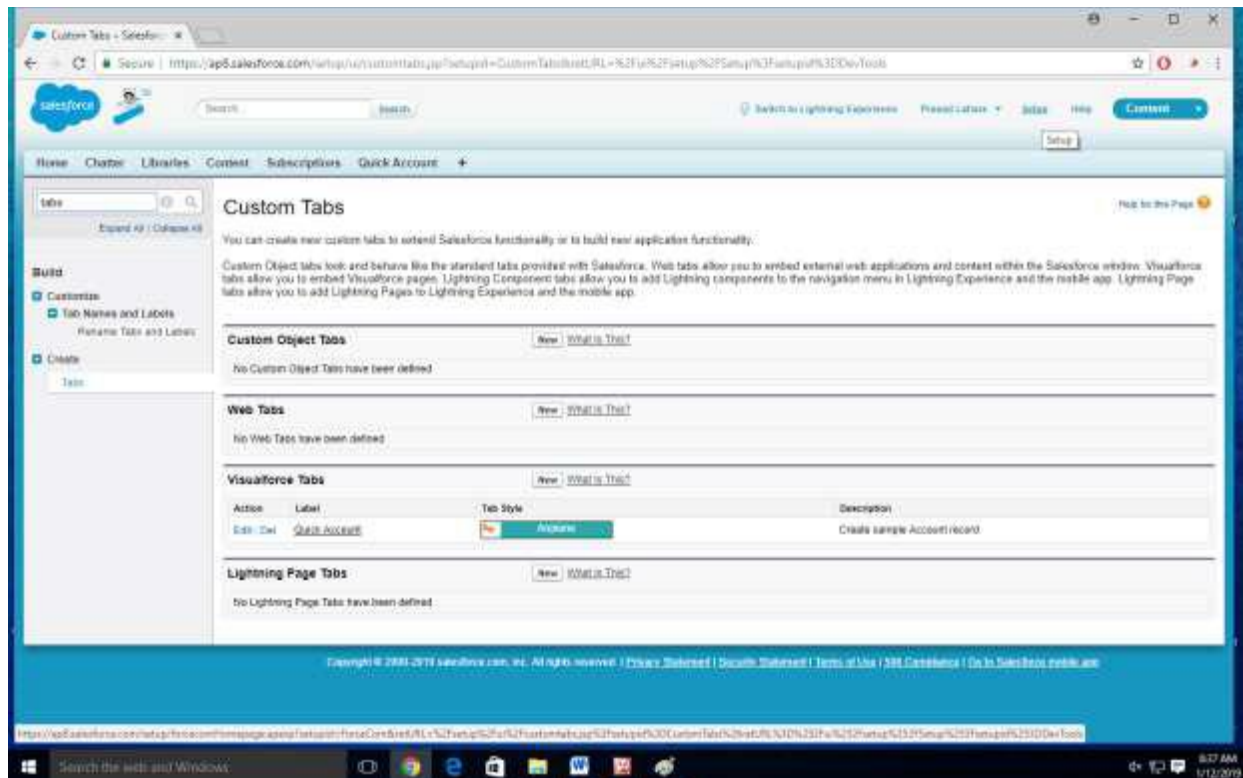


Step No 5:

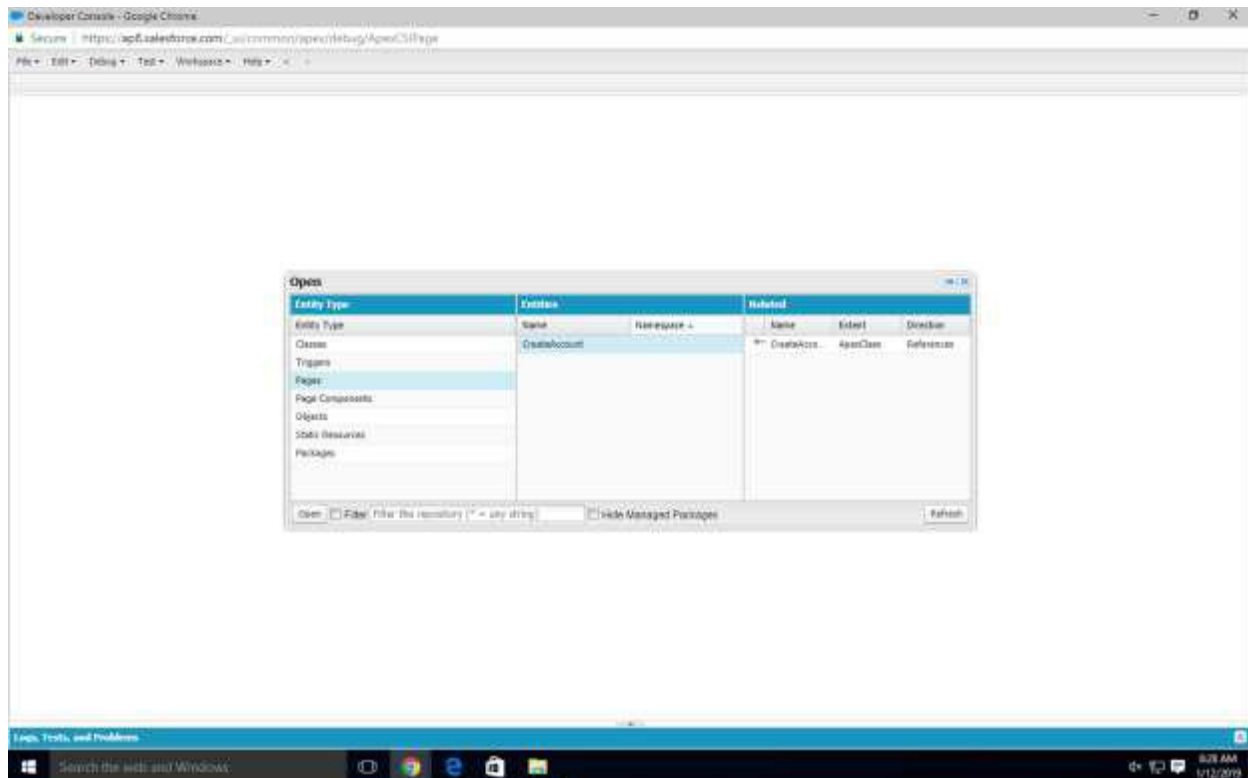
Click on file then Select Apex class and write a code

**Step No 6:**

Select tabs section (for creating GUI)

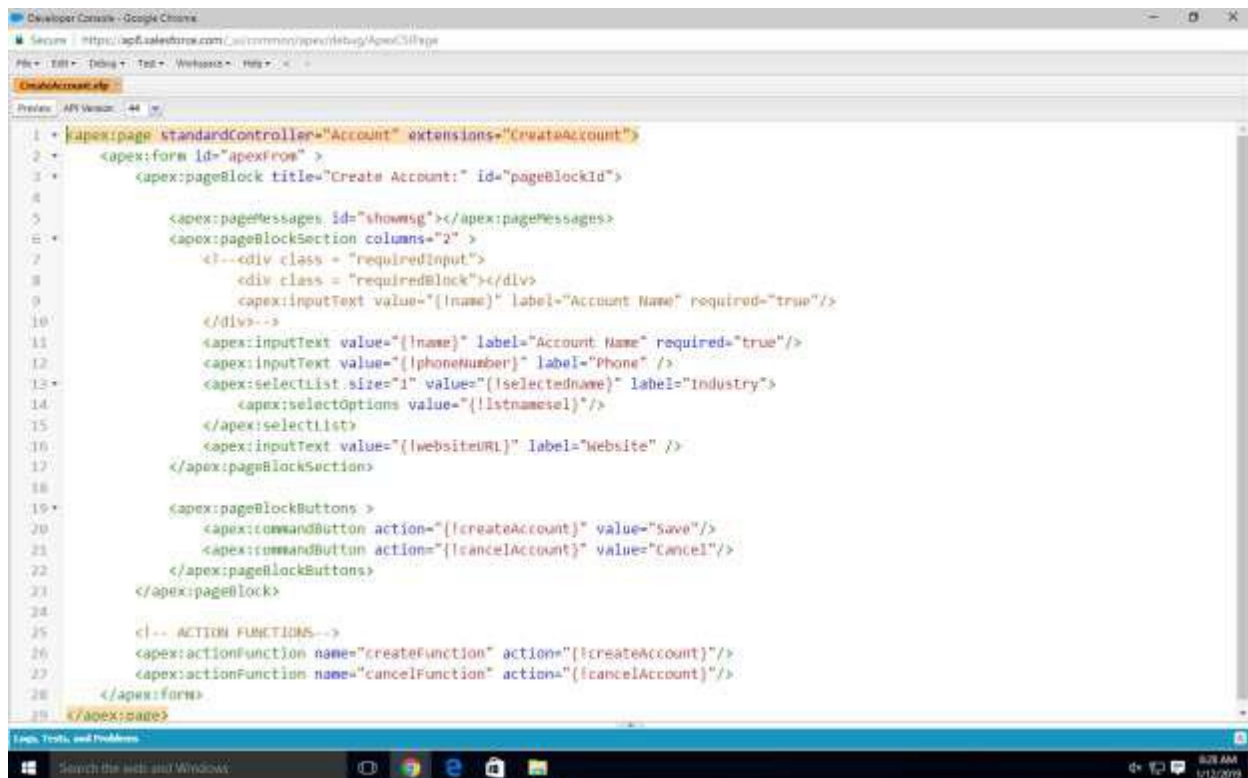


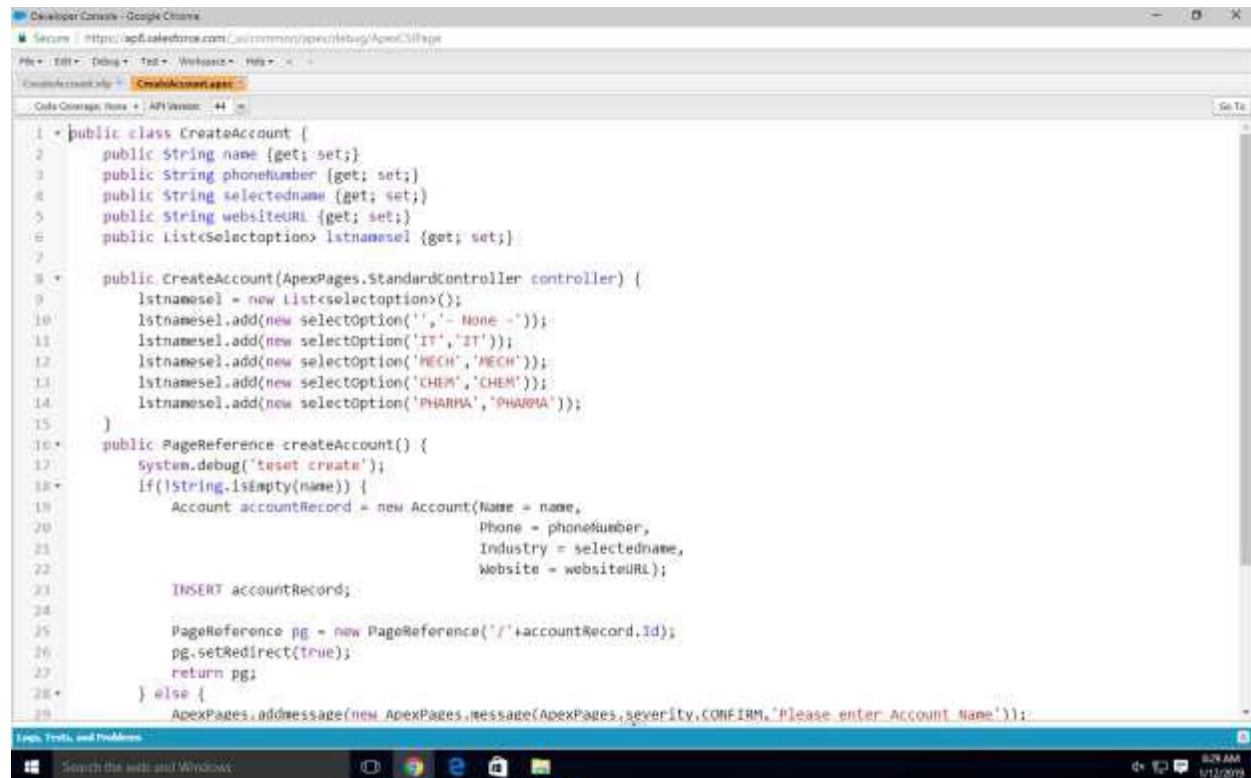
Step No 7: Open a pages



Step No 8:

Program Windows





```

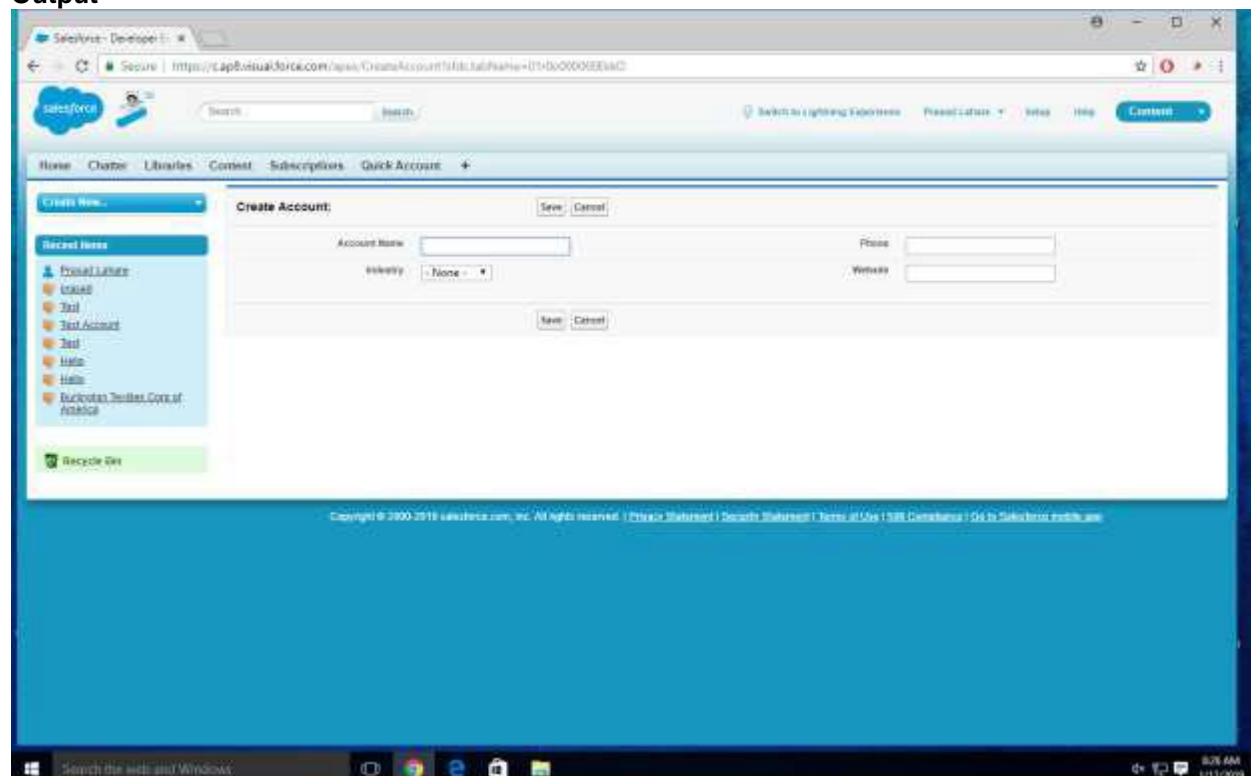
1 public class CreateAccount {
2     public String name {get; set;}
3     public String phoneNumber {get; set;}
4     public String selectedname {get; set;}
5     public String websiteURL {get; set;}
6     public List

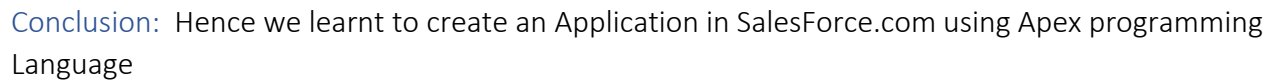
```

Step No 9:

Program Screenshots

Output





Assignment No: 10

Aim: Design and develop custom Application (Mini Project) using Salesforce Cloud.

Theory:Objectives:

1. To design application custom Application.
2. To develop application using sales force cloud

Software Requirements:

Ubuntu
18.04
PHP/python
MySQL
Apex

Hardware Requirements:

Pentium IV system with latest configuration

Introduction

Salesforce.com Inc. is an American cloud-based software company headquartered in San Francisco, California. Though the bulk of its revenue comes from a customer relationship management (CRM) product, Salesforce also sells a complementary suite of enterprise applications focused on customer service, marketing automation, analytics and application development.

Salesforce is the primary enterprise offering within the Salesforce platform. It provides companies with an interface for case management and task management, and a system for automatically routing and escalating important events. The Salesforce customer portal provides customers the ability to track their own cases, includes a social networking plug-in that enables the user to join the conversation about their company on social networking websites, provides analytical tools and other services including email alert, Google search, and access to customers' entitlement and contracts.

Lightning Platform

Lightning Platform (also known as Force.com) is a platform as a service (PaaS) that allows developers to create add-on applications that integrate into the main Salesforce.com application. These third-party applications are hosted on Salesforce.com's infrastructure.

Force.com applications are built using declarative tools, backed by Lightning and Apex (a proprietary Java-like programming language for Force.com) and Lightning and Visual force (a framework that includes an XML syntax typically used to generate HTML). The Force.com platform typically receives three complete releases a year. As the platform is provided as a service to its developers, every single development instance also receives all these updates.

Community Cloud

Community Cloud provides Salesforce customers the ability to create online web properties for external collaboration, customer service, channel sales, and other custom portals in their instance of Salesforce. Tightly integrated to Sales Cloud, Service Cloud, and App Cloud, Community Cloud can be quickly customized to provide a wide variety of web properties

Salesforce Sales Cloud

Salesforce Sales Cloud is a customer relationship management (CRM) platform designed to support sales, marketing and customer support in both business-to-business (B2B) and business-to-customer (B2C) contexts. Sales Cloud is a fully customizable product that brings all the customer information together in an integrated platform that incorporates marketing, lead generation, sales, customer service and business analytics and provides access to thousands of applications through the AppExchange. The platform is provided as Software as a Service (SaaS) for browser-based access; a mobile app is also available. A real-time social feed for collaboration allows users to share information or ask questions of the user community. Salesforce.com offers five versions of Sales Cloud on a per-user, per month basis, from lowest to highest: Group, Professional, Enterprise, Unlimited and Performance. The company offers three levels of support contracts: Standard Success Plan, Premier Success Plan and Premier+ Success Plan.

Create Custom Apps for Salesforce Classic

Create custom apps to give your Salesforce Classic users' access to everything they need all in one place.

If you're new to custom apps, we recommend using Lightning Platform quick start to create an app. With this tool, you can generate a basic working app in just one step.

If you've already created the objects, tabs, and fields you need for your app, follow these steps. With this option, you create an app label and logo, add items to the app, and assign the app to profiles.

1. From Setup, enter Apps in the Quick Find box, then select **Apps**.
2. Click **New**.
3. If the Salesforce console is available, select whether you want to define a custom app

or aSalesforce console.

4. Give the app a name and description.

An app name can have a maximum of 40 characters, including spaces.

5. Optionally, brand your app by giving it a custom logo.

6. Select which items to include in the app.

7. Optionally, set the default landing tab for your new app using the **Default Landing Tab** drop-down menu below the list of selected tabs. This determines the first tab a user sees when logging into this app.

8. Choose which profiles the app will be visible to.

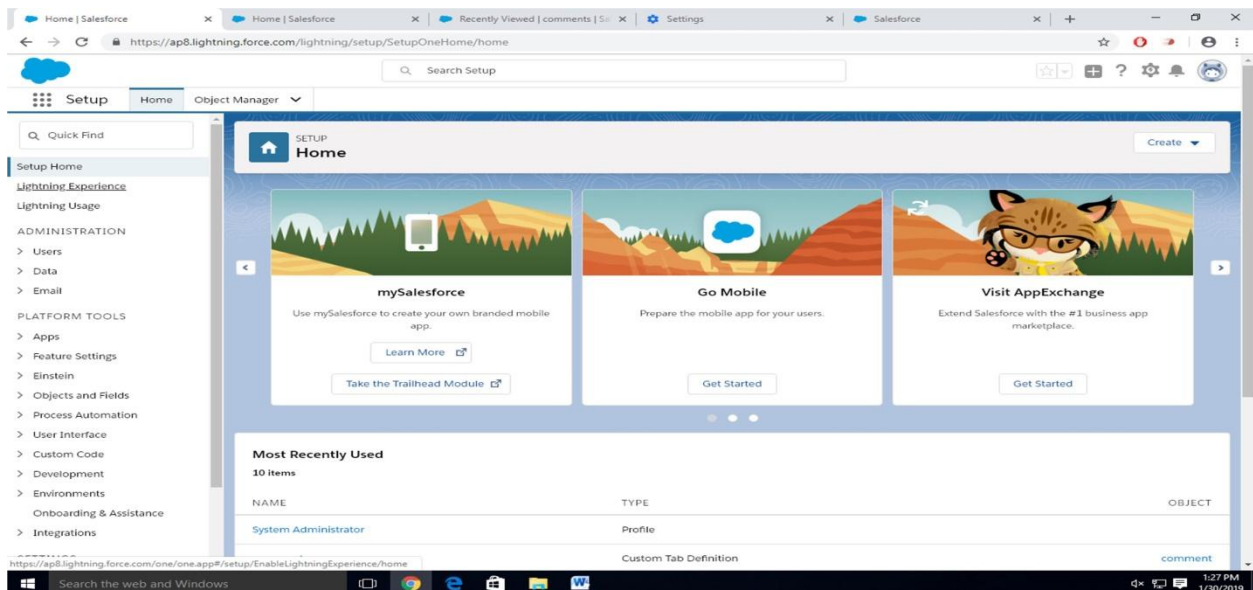
9. Check the Default box to set the app as that profile's default app, meaning that new users with the profile see this app the first time they log in. Profiles with limits are excluded from this list.

10. Click **Save**

Follow the Steps for implimentation

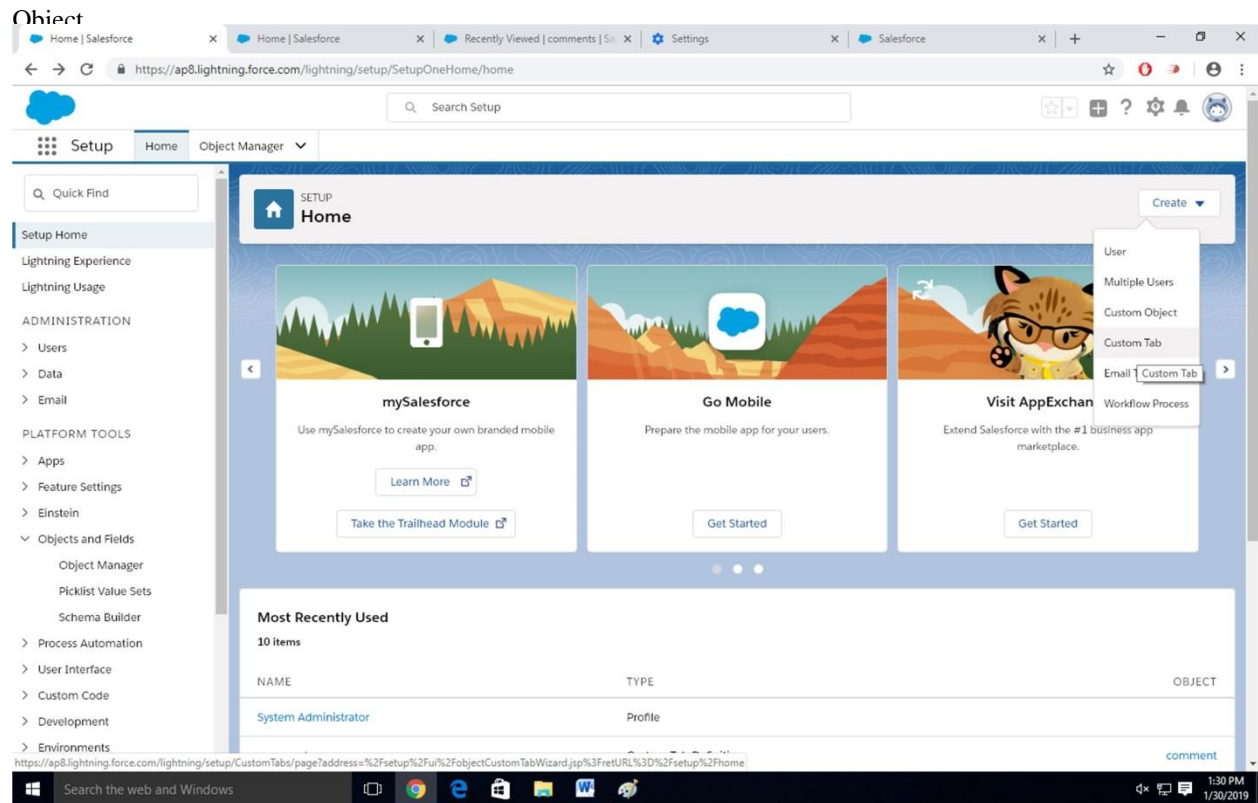
Step-1: Click on Lightning Experience

Step-2: Click on Setup and select Setup for current App.Step-3:



Click on Create an Object

So Click on Object Manager Tab next to Home Tab Click on Create –Custom



Step-4 New custom object page OpenLabel as a-

Comment

Plural label- comments

Object Manager | Salesforce x Home | Salesforce x Recently Viewed | comments | Settings x Salesforce x

https://ap8.lightning.force.com/lightning/setup/ObjectManager/new

Setup Home Object Manager

SETUP New Custom Object

Permissions for this object are disabled for all profiles by default. You can enable object permissions in permission sets or by editing custom profiles. [Tell me more](#) [Don't show this message again](#)

Custom Object Definition Edit

Save Save & New Cancel

Custom Object Information

The singular and plural labels are used in tabs, page layouts, and reports. * = Required Information

Label Example: Account

Plural Label Example: Accounts

Starts with vowel sound ☐

The Object Name is used when referencing the object via the API

Object Name Example: Account

Description

Context-Sensitive Help Setting

☒ Open the standard Salesforce.com Help & Training window

☐ Open a window using a Visualforce page

Content Name

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name Example: Account Name

Data Type

Optional Features

☐ Allow Reports

☐ Allow Activities

☐ Track Field History

☐ Allow in Chatter Groups

Object Classification

When these settings are enabled, this object is classified as an Enterprise Application object. When these settings are disabled, this object is classified as a Light Application object. [Learn more](#)

☒ Allow Sharing

☒ Allow Bulk API Access

☒ Allow Streaming API Access

Deployment Status

☐ In Development

☒ Deployed [What is this?](#)

Search Status

When this setting is enabled, your users can find records of this object type when they search. [Learn more](#)

☐ Allow Search

Object Creation Options (Available only when custom object is first created)

Give Record Name as –comment nameData type- text

Select Allow Reports Check BoxClick on Save

Object Manager | Salesforce x Home | Salesforce x Recently Viewed | comments | Settings x Salesforce x

https://ap8.lightning.force.com/lightning/setup/ObjectManager/new

Setup Home Object Manager

SETUP New Custom Object

Permissions for this object are disabled for all profiles by default. You can enable object permissions in permission sets or by editing custom profiles. [Tell me more](#) [Don't show this message again](#)

Custom Object Definition Edit

Save Save & New Cancel

Custom Object Information

The singular and plural labels are used in tabs, page layouts, and reports. * = Required Information

Label Example: Account

Plural Label Example: Accounts

Starts with vowel sound ☐

The Object Name is used when referencing the object via the API

Object Name Example: Account

Description

Context-Sensitive Help Setting

☒ Open the standard Salesforce.com Help & Training window

☐ Open a window using a Visualforce page

Content Name

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name Example: Account Name

Data Type

Optional Features

☒ Allow Reports

☐ Allow Activities

☐ Track Field History

☐ Allow in Chatter Groups

Object Classification

When these settings are enabled, this object is classified as an Enterprise Application object. When these settings are disabled, this object is classified as a Light Application object. [Learn more](#)

☒ Allow Sharing

☒ Allow Bulk API Access

☒ Allow Streaming API Access

Deployment Status

☐ In Development

☒ Deployed [What is this?](#)

Search Status

When this setting is enabled, your users can find records of this object type when they search. [Learn more](#)

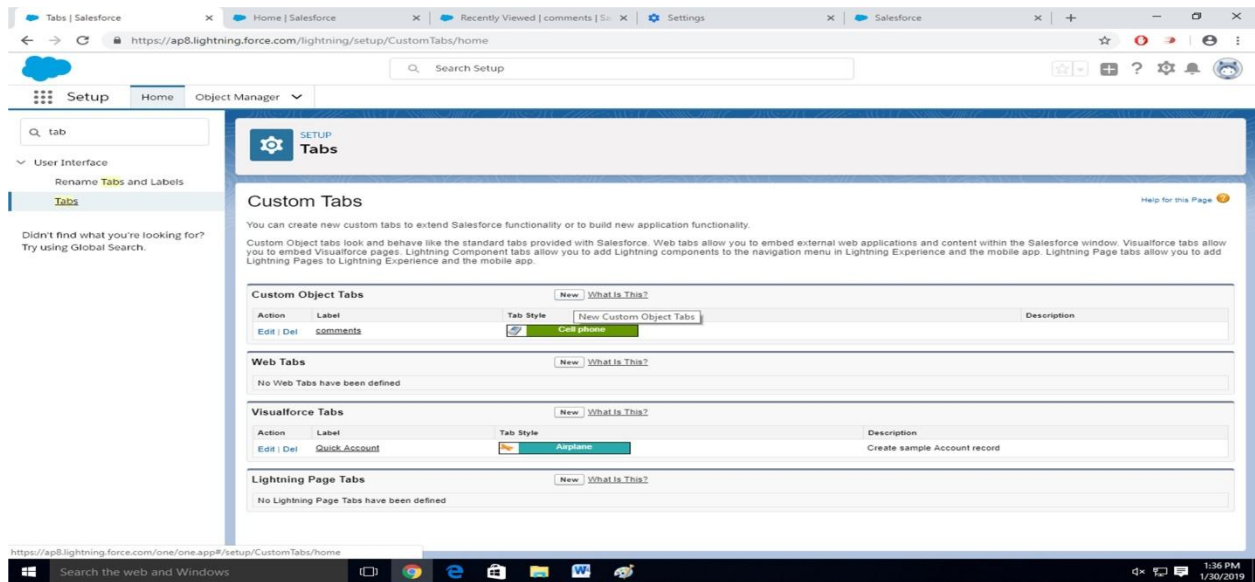
☐ Allow Search

Object Creation Options (Available only when custom object is first created)

Step-5

Click on Home-Search Tabs in Quick searchSelect Custom

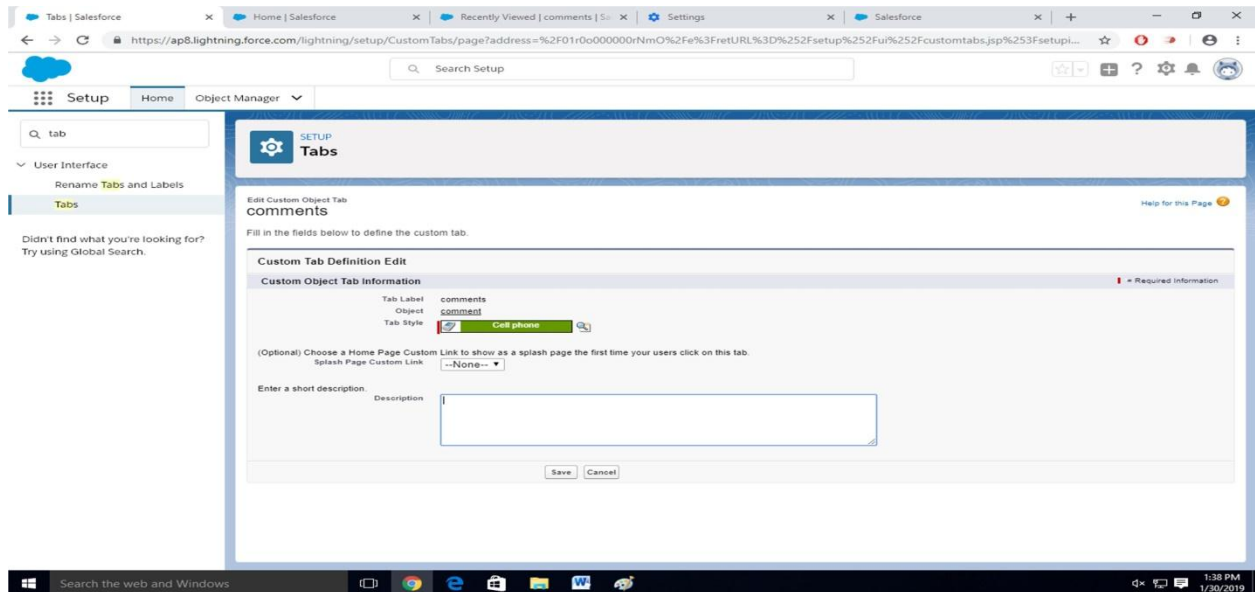
Object-Click on New



Step-6

For Object Select Comment For Tab Style

Select Any Icon



Click-Next-Next-Save

Step-7

Search App Manager in Quick Search and select app manager

App Manager | Salesforce x Home | Salesforce x Recently Viewed | comments | Settings x Salesforce x

https://ap8.lightning.force.com/lightning/setup/NavigationMenus/home

Setup Home Object Manager

Search Setup

app mana

Apps

App Manager

Didn't find what you're looking for? Try using Global Search.

SETUP Lightning Experience App Manager

New Lightning App New Connected App

New Lightning App

16 Items • Sorted by App Name • Filtered by all appmenuitems - TabSet Type

	APP NAME	DEVELOPER NAME	DESCRIPTION	LAST MODIFIED	APP...	VI...
1	Analytics Studio	Insights		1/11/2019 1:34 AM	Classic	✓
2	App Launcher	AppLauncher	App Launcher tabs	1/11/2019 1:34 AM	Classic	✓
3	Bolt Solutions	LightningBolt	Discover and manage business solutions designed for your industry.	1/11/2019 1:34 AM	Lightning	✓
4	comment box	comment_box		1/29/2019 11:41 PM	Lightning	✓
5	Community	Community	Salesforce CRM Communities	1/11/2019 1:34 AM	Classic	✓
6	Content	Content	Salesforce CRM Content	1/11/2019 1:34 AM	Classic	✓
7	Lightning Usage App	LightningInstrumentation	View Adoption and Usage Metrics for Lightning Experience	1/11/2019 1:34 AM	Lightning	✓
8	Marketing	Marketing	Best-in-class on-demand marketing automation	1/11/2019 1:34 AM	Classic	✓
9	Platform	Platform	The fundamental Lightning Platform	1/11/2019 1:34 AM	Classic	✓
10	Sales	Sales	The world's most popular sales force automation (SFA) solution	1/11/2019 1:34 AM	Classic	✓
11	Sales	LightningSales	Manage your sales process with accounts, leads, opportunities, and more	1/11/2019 1:34 AM	Lightning	✓
12	Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multiple records on on...	1/11/2019 1:34 AM	Lightning	✓
13	Salesforce Chatter	Chatter	The Salesforce Chatter social network, including profiles and feeds	1/11/2019 1:34 AM	Classic	✓
14	Service	Service	Manage customer service with accounts, contacts, cases, and more	1/11/2019 1:34 AM	Classic	✓
15	Service Console	LightningService	(Lightning Experience) Lets support agents work with multiple records a...	1/11/2019 1:34 AM	Lightning	✓
16	Site.com	Sites	Build pixel-perfect, data-rich websites using the drag-and-drop Site.com...	1/11/2019 1:34 AM	Classic	✓

Search the web and Windows

1:41 PM 1/30/2019

Enter name to app name

new lightning app

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name ¹

abc

Complete this field.

* Developer Name ¹

Enter a developer name...

Description ¹

Enter a description...

App Branding

Image ¹

Primary Color Hex Value

#0070D2

Upload

Org Theme Options

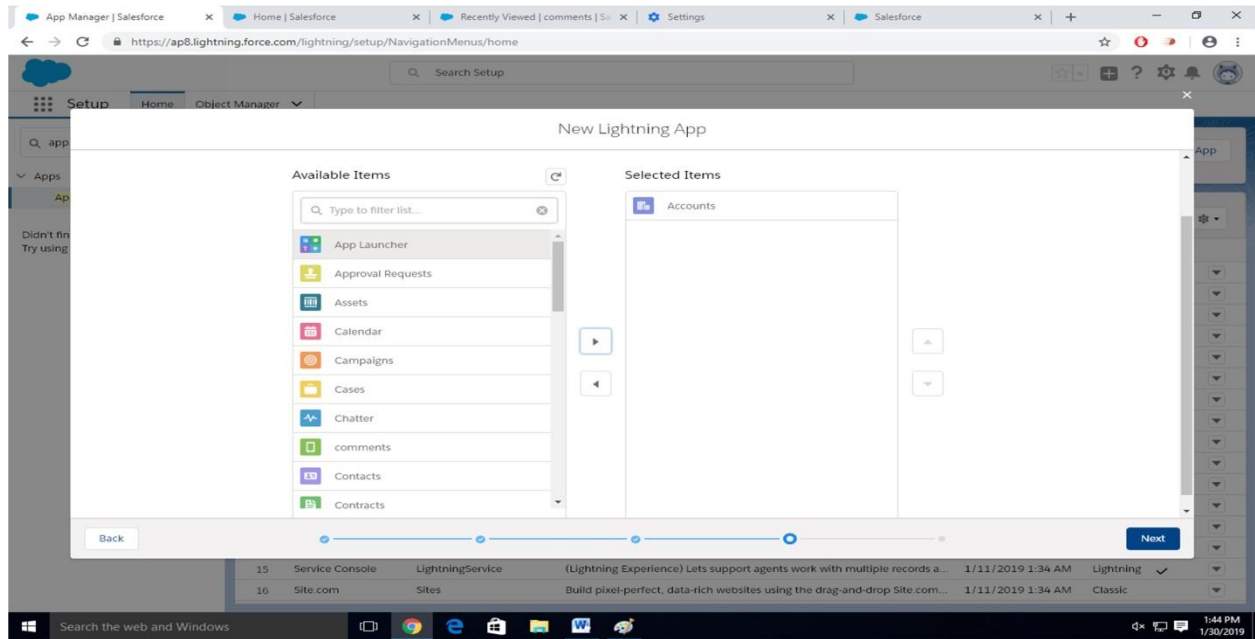
☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

Next

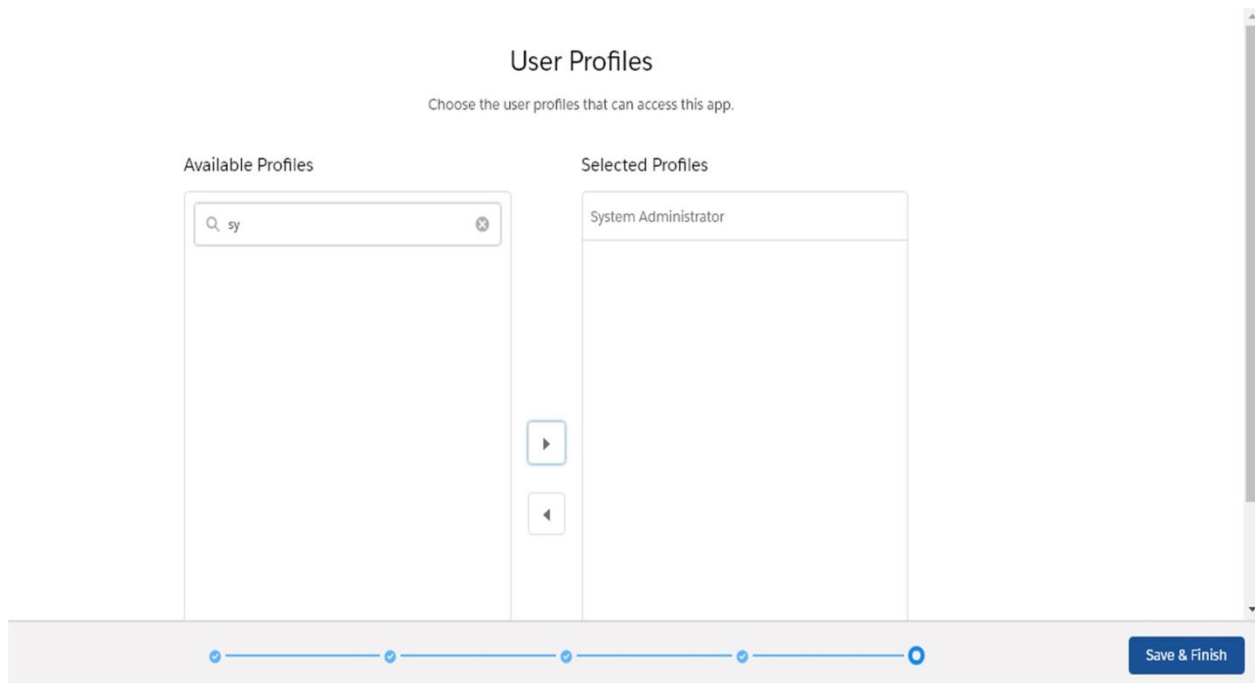
Click on Next-Next-Next.

Select Items (Contacts,Comment)Click on Next




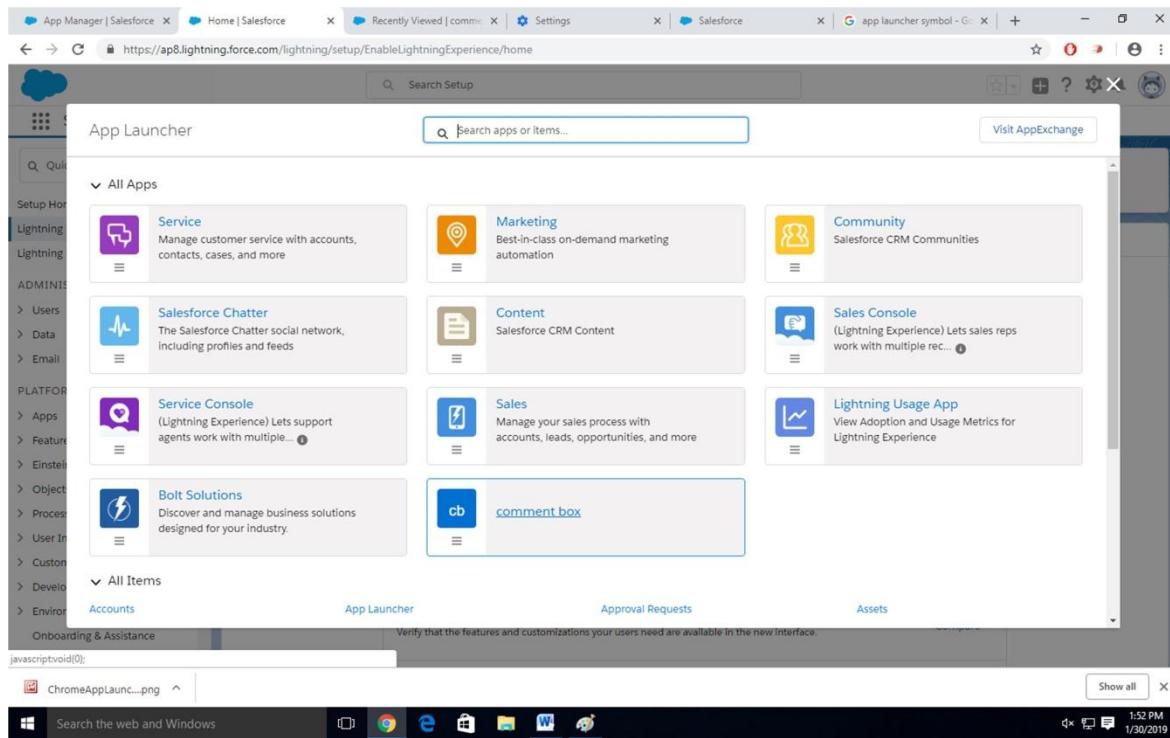
Step-8

Select Profiles (System Administrator) and move to selected profile.Click on Save and Finish.

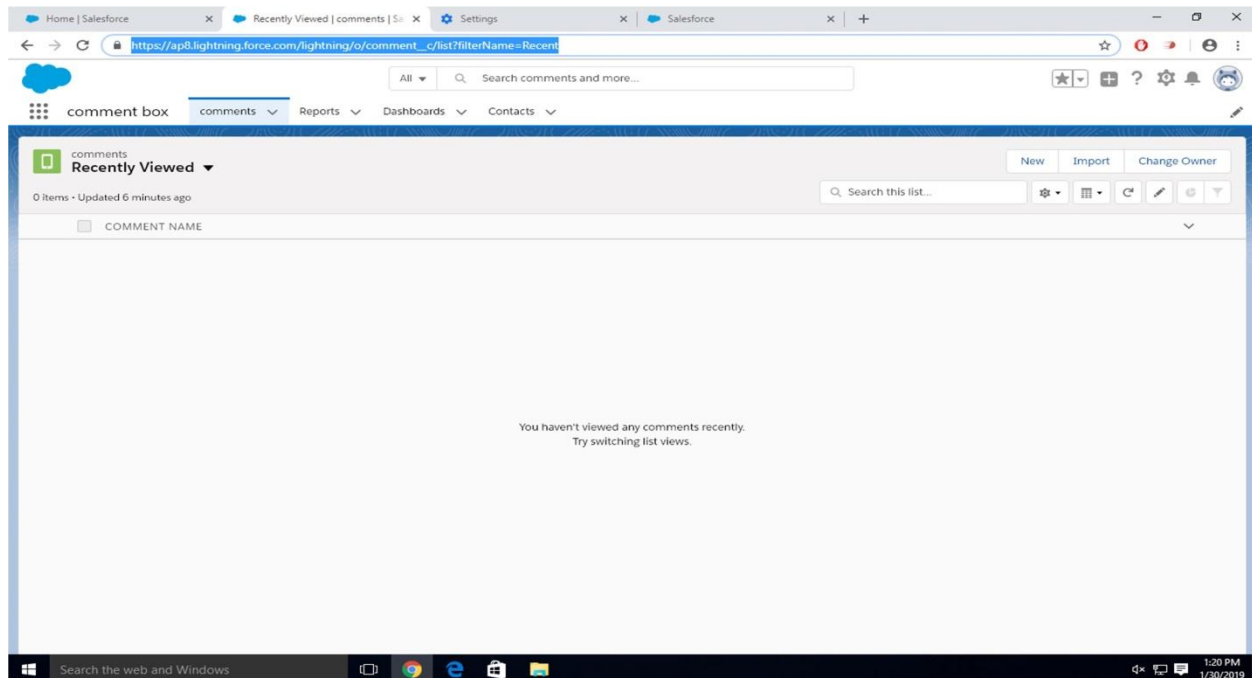


Step-9

Click on App Launcher  Symbol and Select Comment Box App



Step-11 Tour the app



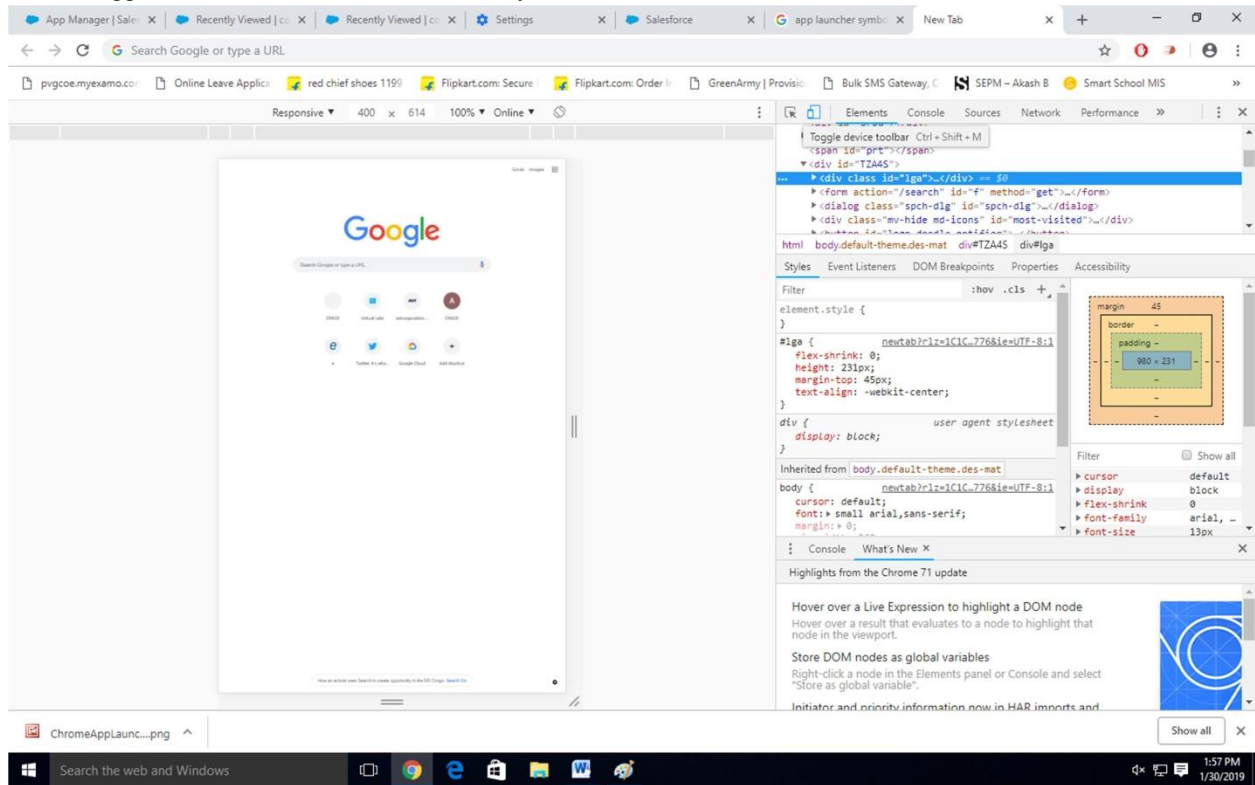
Step-12

Try out mobile app

-Select Chrome developer tools

-Open Chrome-Right Click on Chrome page- Select Inspect

-Click Toggle Device Mode Button to simulate your browser as a mobile device



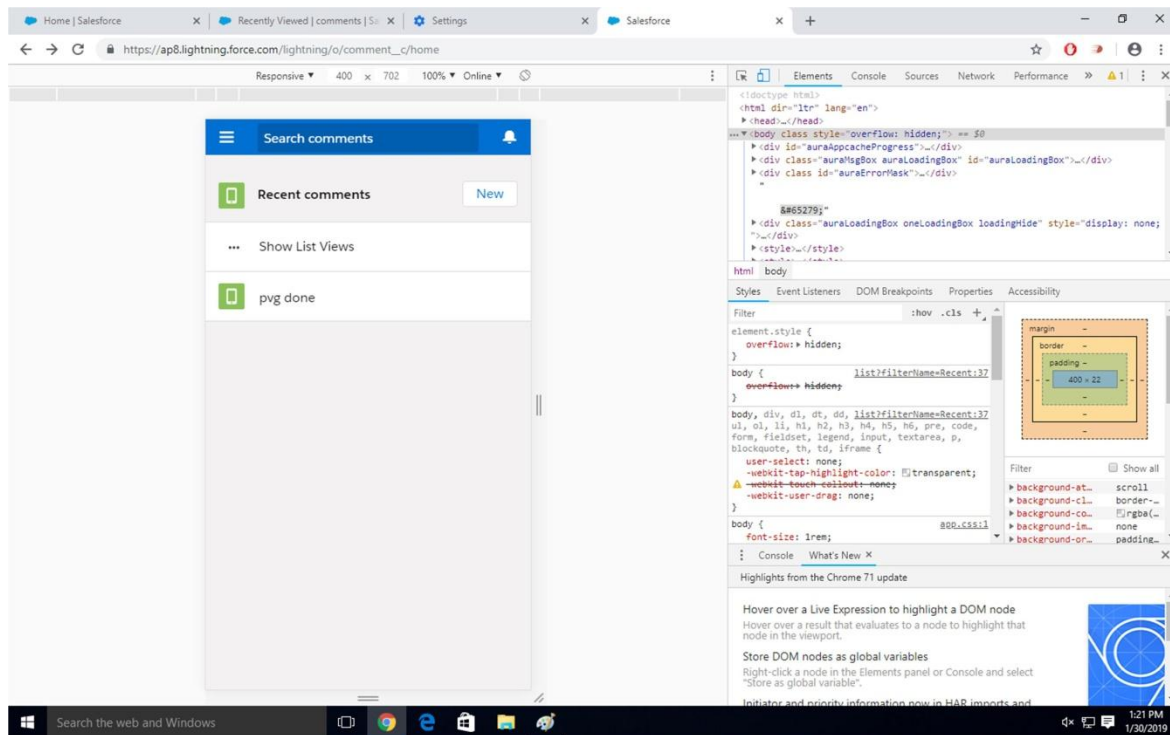
Step-13

To simulate the sales force mobile app in your browser, copy and paste in url from previous tab. Delete the part of the url immediately.

-Click on Left navigation bar

-Find comment object under recent and click on it

-Click new to create a comment



What is the difference between custom application and console application in sales force?

A custom application is a collection of tabs, objects etc that function together to solve a particular problem.

A console application uses a specific Salesforce UI - the console. Console applications are intended to enhance productivity by allowing everything to be done from a single, tabbed, screen.

Conclusion: Hence we learnt to design and develop custom Application (Mini Project) using Sales force Cloud

Mini-Project : Setup your own cloud for Software as a Service (SaaS) over the existing

LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open source technologies to implement **with HDFS**. Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form.

<https://www.tecmint.com/openstack-installation-guide-rhel-centos/>

To set static IP address

<http://www.mustbegeek.com/configure-static-ip-address-in-centos/>

<https://www.tecmint.com/create-deploy-and-launch-virtual-machines-in-openstack/>

<https://www.tecmint.com/openstack-networking-guide/>

configure Sahara in Openstack

