



SRTCT'S
SUMAN RAMESH TULSIANI TECHNICAL CAMPUS – FACULTY OF ENGINEERING,
KHAMSHET
An ISO 9001:2015 Certified Institute
NAAC Accredited Institute

DEPARTMENT OF COMPUTER ENGINEERING

Academic Year: 2022-23
Semesters -II
LABORATORY MANUAL

Class: TE

Subject: LP-II (2019 Course)[310258]

Subject In charge: Prof. Anjali M. Dalvi

Subject In charge: Prof. Satish S. Asane

SRTTC FOE LABORATORY PRACTICE II



SRTCT'S
SUMAN RAMESH TULSIANI TECHNICAL CAMPUS – FACULTY OF
ENGINEERING, KHAMSHET
An ISO 9001:2015 Certified Institute
NAAC Accredited Institute
DEPARTMENT OF COMPUTER ENGINEERING

INDEX

Department of Computer Engineering

Class: T.E.

S.R. No.	Name of the Experiment	Date of Conduction	Date of Checking	Page No.	Sign	Remarks
1	Software Modeling and Architectures Consider a library, where a member can perform two operations: issue book and return it. A book is issued to a member only after verifying his credentials. Develop a use case diagram for the given library system by identifying the actors and use cases and associate the use cases with the actors by drawing a use case diagram. Use UML tool					
2	Consider online shopping system. Perform the following tasks and draw the class diagram using UML tool. Represent the individual classes, and objects Add Methods Represent relationships and other classifiers like interfaces					
3	Mini-Projects					

4	Artificial Intelligence 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.					
5	Implement Greedy search algorithm for any of the following application: Kruskal's Minimum Spanning Tree Algorithm					
6	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.					
7	Develop an elementary chatbot for any suitable customer interaction application.					
8	Implement any one of the following Expert System I. Information management					



SRTCT'S
**SUMAN RAMESH TULSIANI TECHNICAL CAMPUS – FACULTY OF
ENGINEERING, KHAMSHET**

An ISO 9001:2015 Certified Institute

NAAC Accredited Institute

DEPARTMENT OF COMPUTER ENGINEERING

Academic Year: 2022-23

CERTIFICATE

This is to certify that Mr. /Miss. _____
of Class T.E. Computer Roll No. _____ has satisfactory completed
practical of the subject “Laboratory Practice II Lab” for 2nd semester of Academic
Year 2022 – 2023.

Date:

Prof.A.M.Dalvi

Prof.S.S.Asane

Prof. A. M. Dalvi

Prof. (Dr.) J. B. Sankpal

Subject Teacher

Subject Teacher

Head of Department

Principal

SRTTC FOE LABORATORY PRACTICE II

Experiment No: 1

Name of the Student:

Class: TE Batch: Date of

Performance:

Signature of the Subject Incharge:

Assignment No. 1

Consider a library, where a member can perform two operations: issue book and return it. A book is issued to a member only after verifying his credentials. Develop a use case diagram for the given library system by identifying the actors and use cases and associate the use cases with the actors by drawing a use case diagram. Use UML tool

A Library Management System is a software built to handle the primary housekeeping functions of a library. Libraries rely on library management systems to manage asset collections as well as relationships with their members. Library management systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Library management systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

System Requirements

Always clarify requirements at the beginning of the interview. Be sure to ask questions to find the exact scope of the system that the interviewer has in mind.

We will focus on the following set of requirements while designing the Library Management System:

1. Any library member should be able to search books by their title, author, subject category as well by the publication date.
2. Each book will have a unique identification number and other details including a rack number which will help to physically locate the book.
3. There could be more than one copy of a book, and library members should be able to check-out and reserve any copy. We will call each copy of a book, a book item.
4. The system should be able to retrieve information like who took a particular book or what are the books checked-out by a specific library member.

5. There should be a maximum limit (5) on how many books a member can check-out.
6. There should be a maximum limit (10) on how many days a member can keep a book.
7. The system should be able to collect fines for books returned after the due date.
8. Members should be able to reserve books that are not currently available.
9. The system should be able to send notifications whenever the reserved books become available, as well as when the book is not returned within the due date.
10. Each book and member card will have a unique barcode. The system will be able to read barcodes from books and members' library cards.

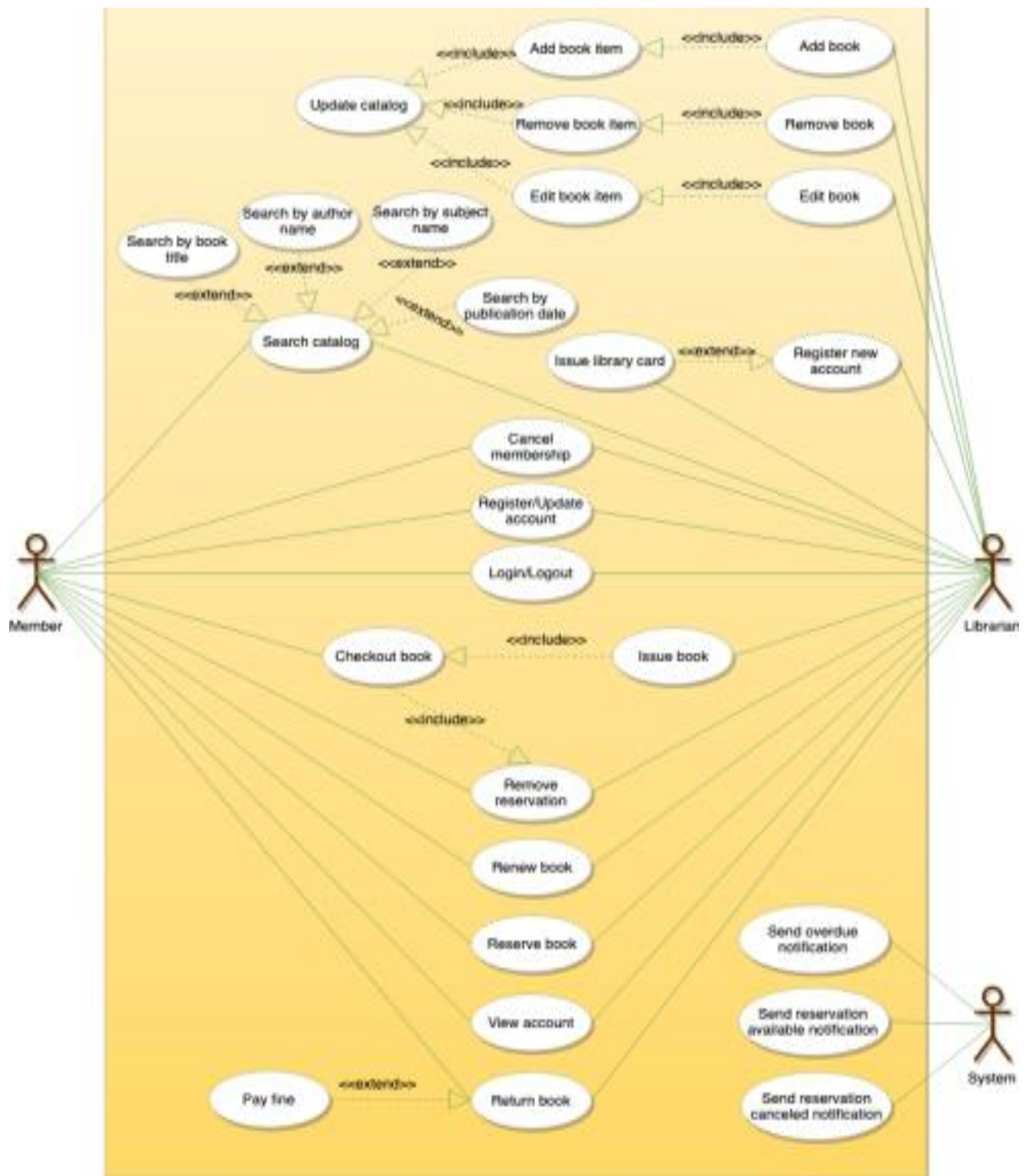
Use case diagram#

We have three main actors in our system:

- **Librarian:** Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
- **Member:** All members can search the catalog, as well as check-out, reserve, renew, and return a book.
- **System:** Mainly responsible for sending notifications for overdue books, canceled reservations, etc.

Here are the top use cases of the Library Management System:

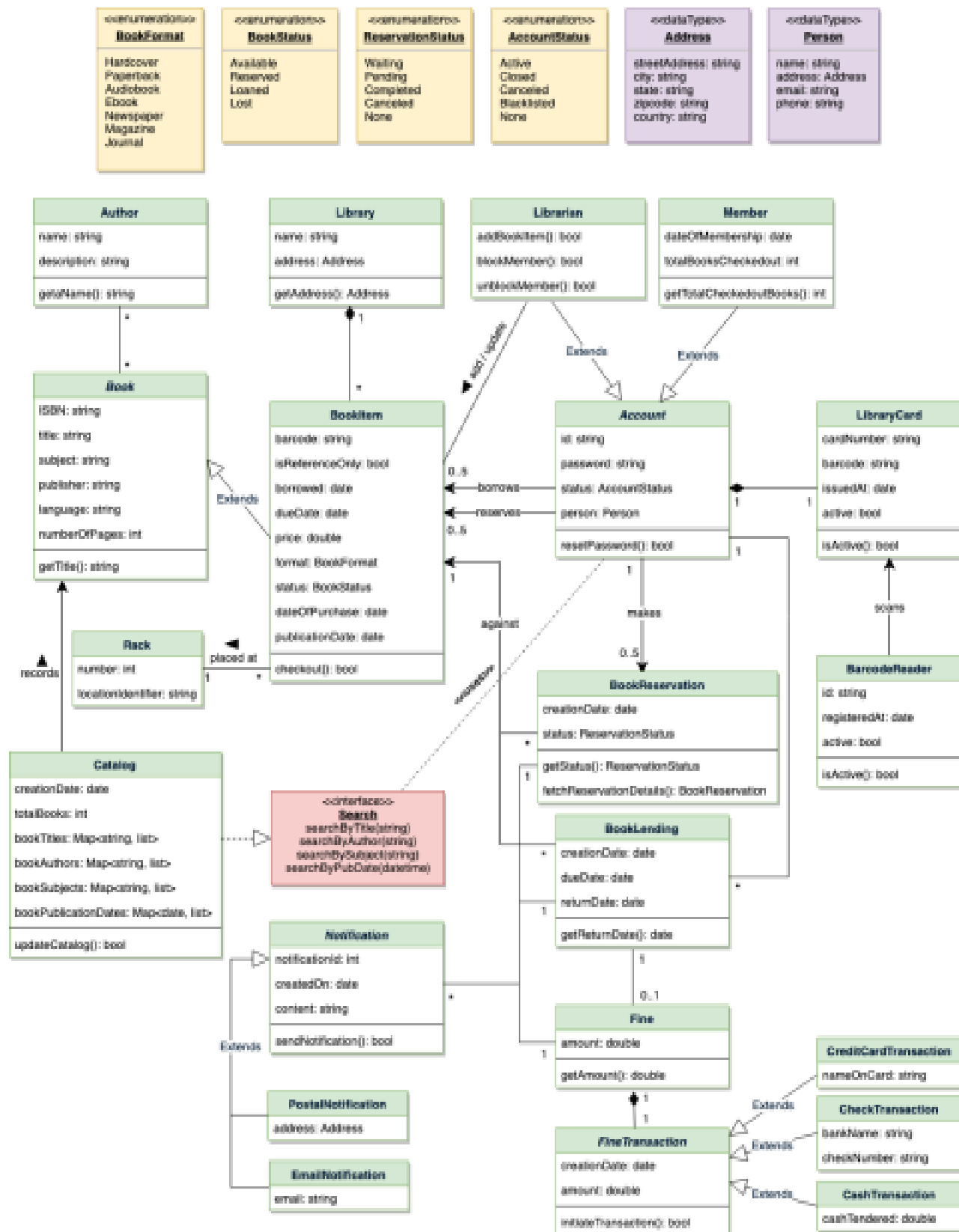
- **Add/Remove/Edit book:** To add, remove or modify a book or book item.
- **Search catalog:** To search books by title, author, subject or publication date.
- **Register new account/cancel membership:** To add a new member or cancel the membership of an existing member.
- **Check-out book:** To borrow a book from the library.
- **Reserve book:** To reserve a book which is not currently available.
- **Renew a book:** To reborrow an already checked-out book.
- **Return a book:** To return a book to the library which was issued to a member.



Class diagram#

Here are the main classes of our Library Management System:

- **Library:** The central part of the organization for which this software has been designed. It has attributes like 'Name' to distinguish it from any other libraries and 'Address' to describe its location.
- **Book:** The basic building block of the system. Every book will have ISBN, Title, Subject, Publishers, etc.
- **BookItem:** Any book can have multiple copies, each copy will be considered a book item in our system. Each book item will have a unique barcode.
- **Account:** We will have two types of accounts in the system, one will be a general member, and the other will be a librarian.
- **LibraryCard:** Each library user will be issued a library card, which will be used to identify users while issuing or returning books.
- **BookReservation:** Responsible for managing reservations against book items.
- **BookLending:** Manage the checking-out of book items.
- **Catalog:** Catalogs contain list of books sorted on certain criteria. Our system will support searching through four catalogs: Title, Author, Subject, and Publish-date.
- **Fine:** This class will be responsible for calculating and collecting fines from library members.
- **Author:** This class will encapsulate a book author.
- **Rack:** Books will be placed on racks. Each rack will be identified by a rack number and will have a location identifier to describe the physical location of the rack in the library.
- **Notification:** This class will take care of sending notifications to library members.

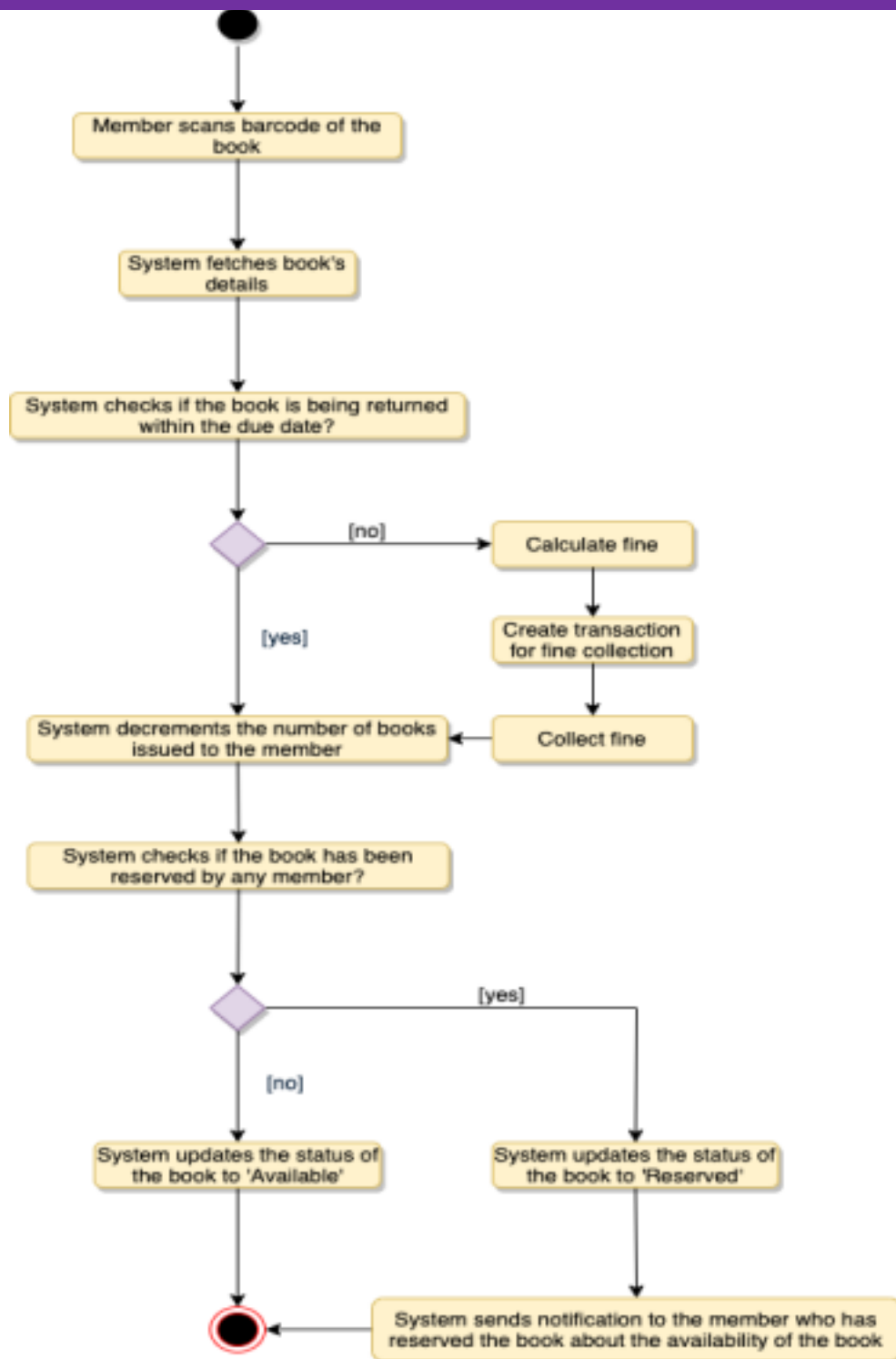


Class diagram for Library Management System

Activity diagrams#

Check-out a book: Any library member or librarian can perform this activity. Here are the set of steps to check-out a book:

Return a book: Any library member or librarian can perform this activity. The system will collect fines from members if they return books after the due date. Here are the steps for returning a book:



Renew a book: While renewing (re-issuing) a book, the system will check for fines and see if any other member has not reserved the same book, in that case the book item cannot be renewed. Here are the different steps for renewing a book:

Experiment No: 2

Name of the Student:

Class: TE Batch: Date of

Performance:

Signature of the Subject Incharge:

Assignment No. 2

Consider online shopping system. Perform the following tasks and draw the class diagram using UML tool. Represent the individual classes, and objects Add Methods Represent relationships and other classifiers like interfaces

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

How to Draw a Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. It is very important to learn the drawing procedure of class diagram.

Class diagrams have a lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system.

The following points should be remembered while drawing a class diagram –

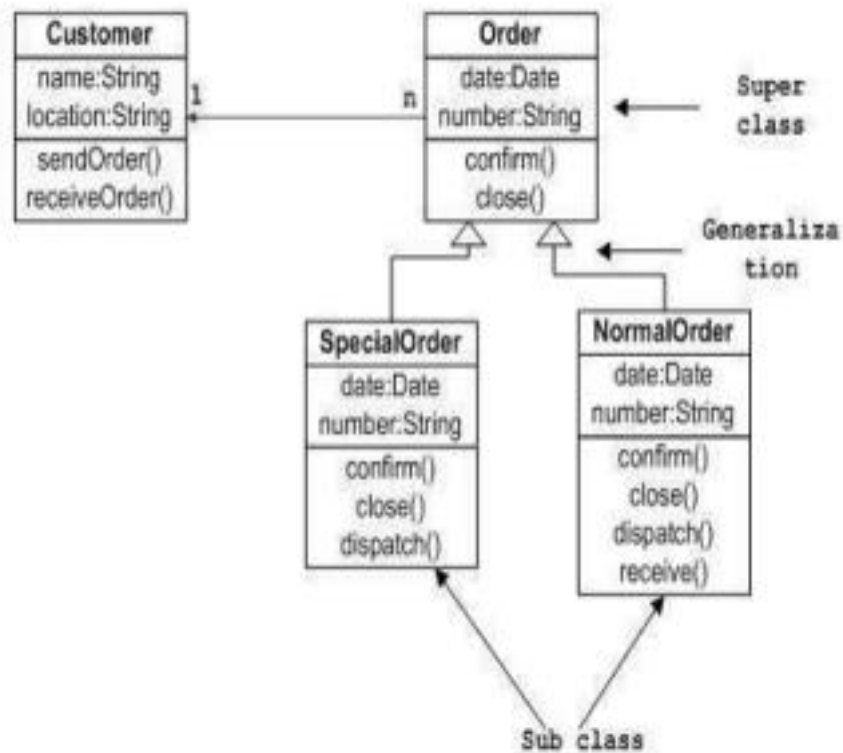
- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

- First of all, Order and Customer are identified as the two elements of the system. They have a one-to-many relationship because a customer can have multiple orders.
- Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().

The following class diagram has been drawn considering all the points mentioned above.

Sample Class Diagram



Where to Use Class Diagrams?

Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.

Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.

Class diagram clearly shows the mapping with object-oriented languages such as Java, C++, etc. From practical experience, class diagram is generally used for construction purpose.

In a nutshell it can be said, class diagrams are used for –

- Describing the static view of the system.
- Showing the collaboration among the elements of the static view.

·Describing the functionalities performed by the system. ·Construction
of software applications using object oriented languages.

Experiment No: 3

Name of the Student:

Class: TE Batch: Date of

Performance:

Signature of the Subject Incharge:

Assignment No. 3

Mini project Problem statement

Select a moderately complex system and narrate concise requirement Specification for the same. Design the system indicating system elements organizations using applicable architectural styles and design patterns with the help of a detailed Class diagram depicting logical architecture. Specify and document the architecture and design pattern with the help of templates. Implement the system features and judge the benefits of the design patterns accommodated

Specific Objectives:

- To create a system that stores data of hospital easily and user friendly.
- Make the entire process of itinerary decisions On-The-Go.
- To provide the customer more options like view doctors, departments ,nurse etc.
- Reduce the workload of searching.
- To enable bulk amount of data search in a very short time we are adding search bar.
- Instant search options for patients.

1.4 Requirements

Purpose

The purpose of this document is to describe the requirements that are necessary for developing the Hospital Management System. It also helps to gather and analyze the new ideas incorporated to the existing Management system. The intended audience is any person who wants to view, store or manage the logs.

1.2 Scope

The system provides an online interface where users can search for specific doctors from any source to destination. The user can view different departments of hospital and specific doctors appointed for it. There are several features offered in this system that sets it apart from the many existing systems, such as allowing to store patients data.

1.3 Objective

General Objective:

The main goal of the proposed system is to make the process of hospital Management online rather than maintaining registers.

1.4.1 Business Requirements

- Investment, to host and advertise the application
- A technical and managerial team, to maintain the system and update it on a consistent basis

1.4.2 System Requirements

- PHP 5.3+, HTML5, Twitter Bootstrap v3, jQuery 1.7, MySQL DB
- Windows/Fedora operating system
- Server Bandwidth : 100TB, Space: unlimited
- Database space: Unlimited

1.4.3 Functional Requirements

The functional requirements of the system are divided among the Customers and Administrator of the application. It can be explained in detail:

1.4.3.1 Use case name: User Registration

- Description: This use case describes the scenario where the customer should provide all their details in order to create an account for themselves to use for login purposes.
- Actor: User or the customer.
- Input: Personal details such as Name, Address, Phone Number and Email ID etc.

- Output: All the details will be saved into a database and can be used to fill in the appropriate fields at the time of signup.

Use Case: Fig 1.1



1.4.3.2 Use case name: User Login/Logout

- Description: This use case describes the scenario where the customer logs into their account by entering the username and password provided at the time of registration.

Experiment No: 4

Name of the Student: _____

Class: TE Batch: _____

Date of Performance: _____

Signature of the Subject Incharge: _____

Assignment No. 4

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.

Depth First Search or DFS for a Graph

Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a Boolean visited array.

Example:

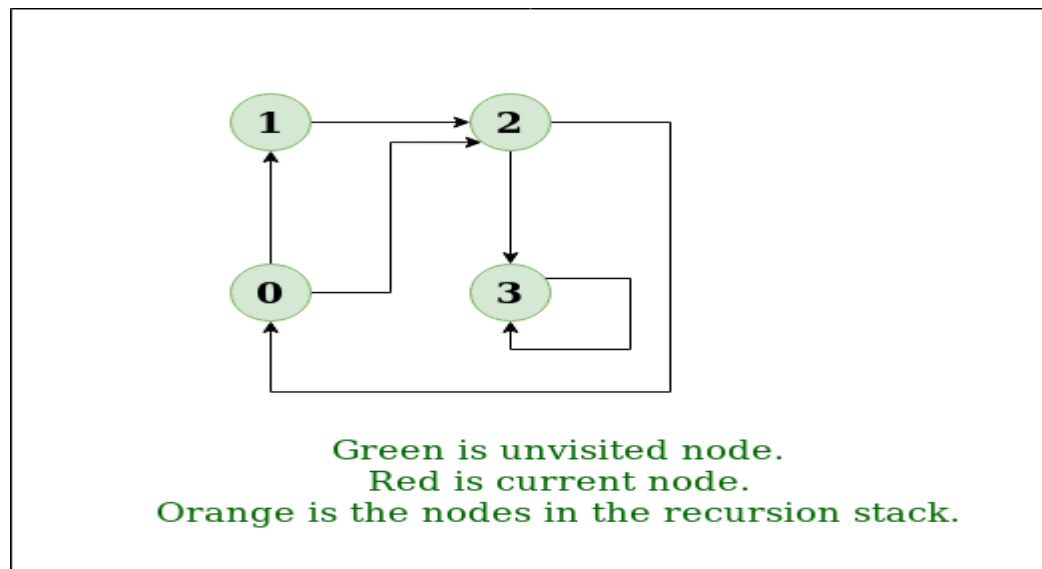
Input: $n = 4, e = 6$

$0 \rightarrow 1, 0 \rightarrow 2, 1 \rightarrow 2, 2 \rightarrow 0, 2 \rightarrow 3, 3 \rightarrow 3$

Output: DFS from vertex 1: 1 2 0 3

Explanation:

DFS Diagram:



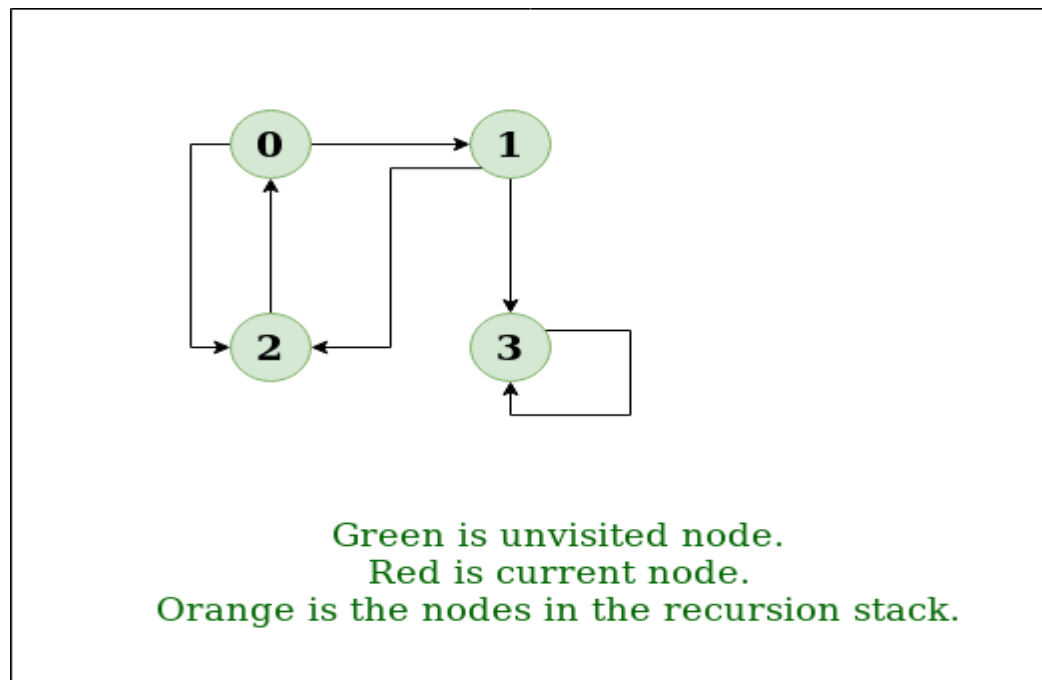
Input: $n = 4, e = 6$

$2 \rightarrow 0, 0 \rightarrow 2, 1 \rightarrow 2, 0 \rightarrow 1, 3 \rightarrow 3, 1 \rightarrow 3$

Output: DFS from vertex 2 : 2 0 1 3

Explanation:

DFS Diagram:



Approach:

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.

Algorithm:

Create a recursive function that takes the index of the node and a visited array.

1. Mark the current node as visited and print the node.
2. Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.

Implementation:

Below are implementations of simple Depth First Traversal. The C++ implementation uses an adjacency list representation of graphs. STL's list container is used to store lists of adjacent nodes.

**// C++ program to print DFS traversal from
// a given vertex in a given graph**

```
#include <bits/stdc++.h>
using namespace std;

// Graph class represents a directed graph
// using adjacency list representation
class Graph {
public:
    map<int, bool> visited;
    map<int, list<int> > adj;

    // function to add an edge to graph
    void addEdge(int v, int w);

    // DFS traversal of the vertices
    // reachable from v
    void DFS(int v);
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w); // Add w to v's list.
}

void Graph::DFS(int v)
{
    // Mark the current node as visited and
    // print it
    visited[v] = true;
    cout << v << " ";

    // Recur for all the vertices adjacent
    // to this vertex
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFS(*i);
}
```

```

// Driver code
int main()
{
    // Create a graph given in the above diagram
    Graph g;
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Depth First Traversal"
    " (starting from vertex 2) \n";
    g.DFS(2);

    return 0;
}

```

Output:

Following is Depth First Traversal (starting from vertex 2) 2 0 1 3

Experiment No: 5

Name of the Student: _____

Class: TE Batch: _____

Date of Performance: _____

Signature of the Subject Incharge: _____

Assignment No. 5

Implement Greedy search algorithm for any of the following

application: Kruskal's Minimum Spanning Tree Algorithm

What is Minimum Spanning Tree?

Given a connected and undirected graph, a *spanning tree* of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A *minimum spanning tree (MST)* or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

How many edges does a minimum spanning tree has?

A minimum spanning tree has $(V - 1)$ edges where V is the number of vertices in the given graph.

Below are the steps for finding MST using Kruskal's algorithm

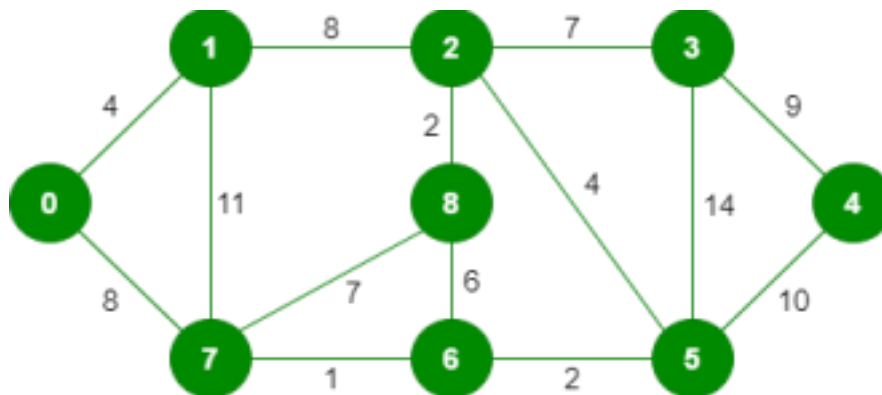
- 1. Sort all the edges in non-decreasing order of their weight.*
- 2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.*
- 3. Repeat step#2 until there are $(V-1)$ edges in the spanning tree.*

Step #2 uses the Union-Find algorithm to detect cycles. So we recommend reading the following post as a prerequisite.

Union-Find Algorithm | Set 1 (Detect Cycle in a Graph)

Union-Find Algorithm | Set 2 (Union By Rank and Path Compression)

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example: Consider the below input graph. weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example: Consider the below input graph.



The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

After sorting:

Weight Src Dest

1 7 6

2 8 2

2 6 5

4 0 1

4 2 5

6 8 6

7 2 3

7 7 8

8 0 7

8 1 2

9 3 4

10 5 4

11 1 7

14 3 5

Now pick all edges one by one from the sorted list of edges

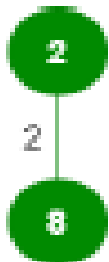
1. *Pick edge 7-6:* No cycle is formed, include it.



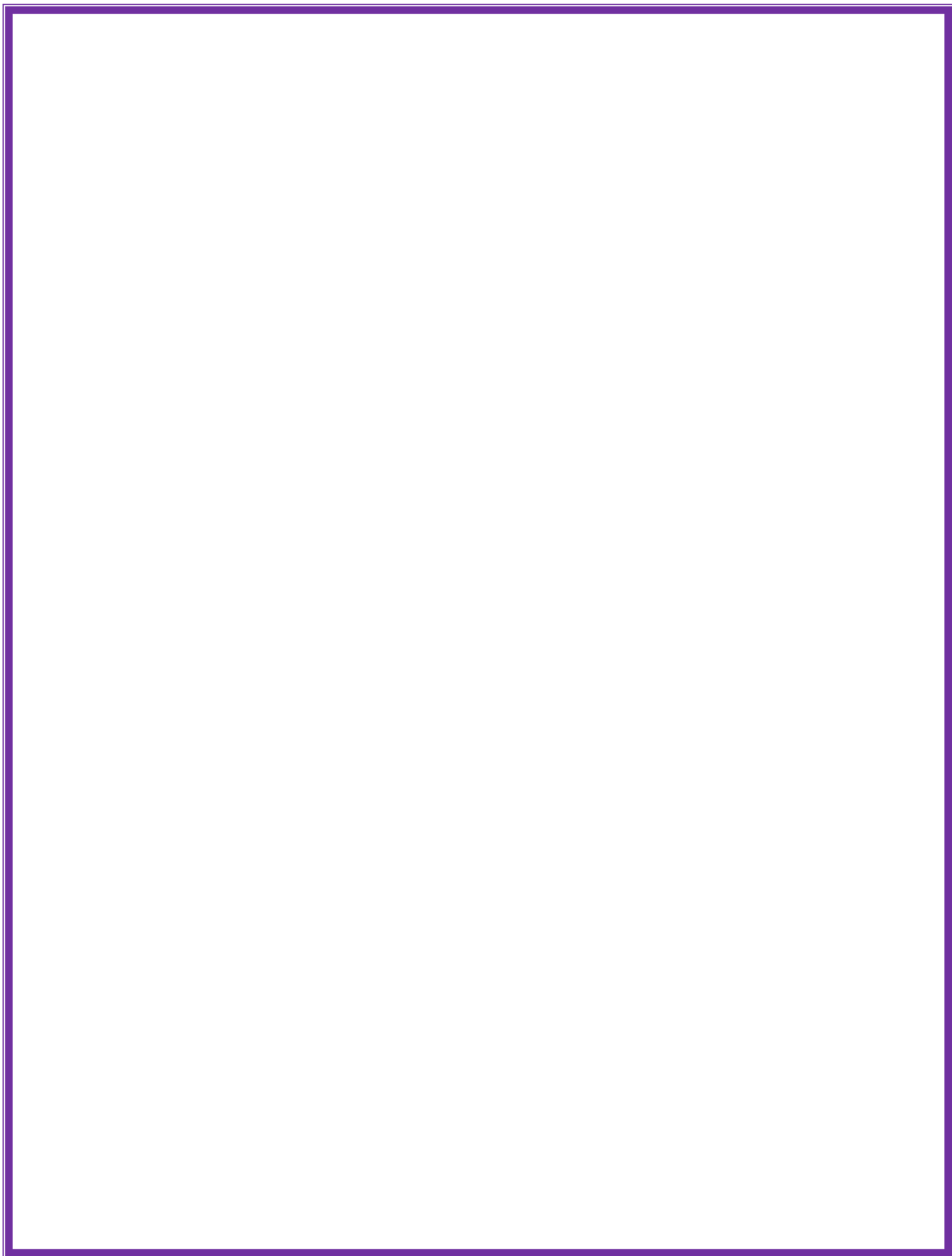
2. *Pick edge 8-2:* No cycle is formed, include it.

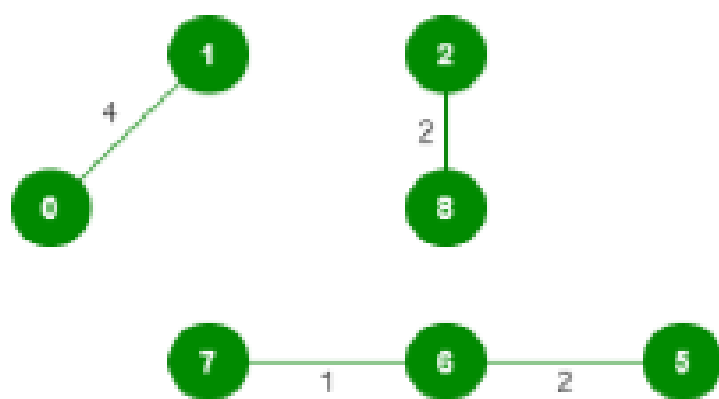


3. *Pick edge 6-5:* No cycle is formed, include it.

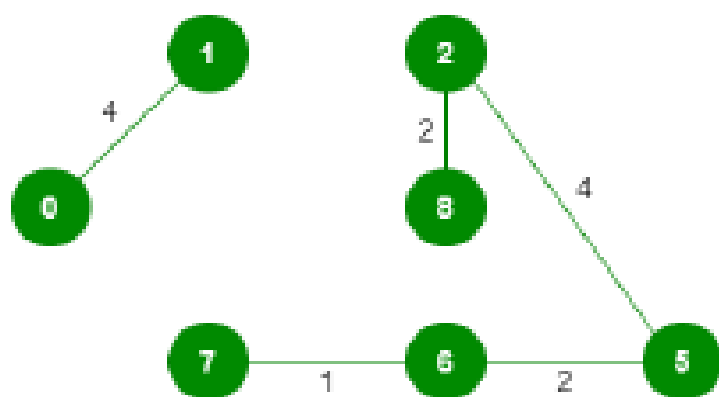


4. *Pick edge 0-1:* No cycle is formed, include it.



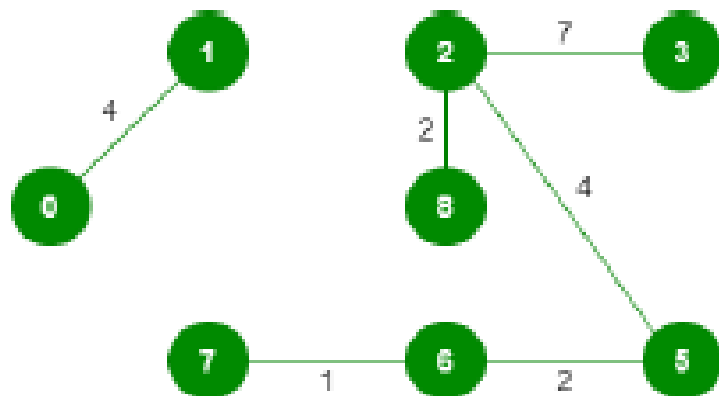


5. Pick edge 2-5: No cycle is formed, include it.



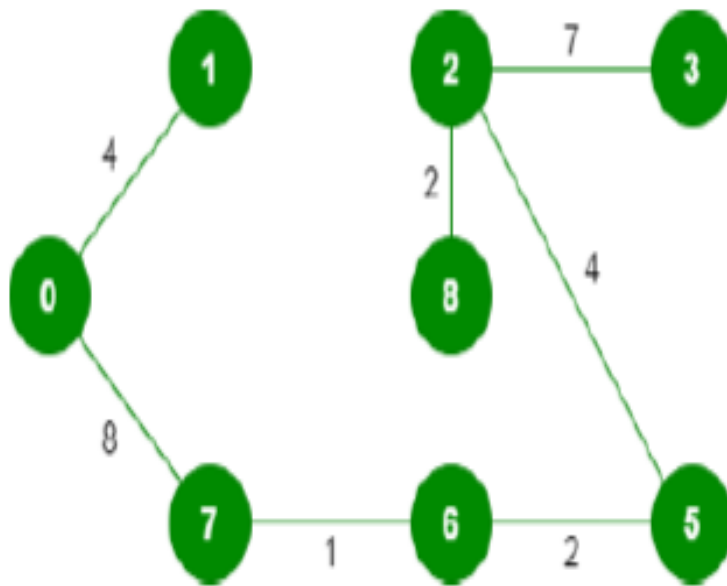
6. Pick edge 3-6: Since including this edge results in the cycle, discard it.

7. Pick edge 2-3: No cycle is formed, include it.



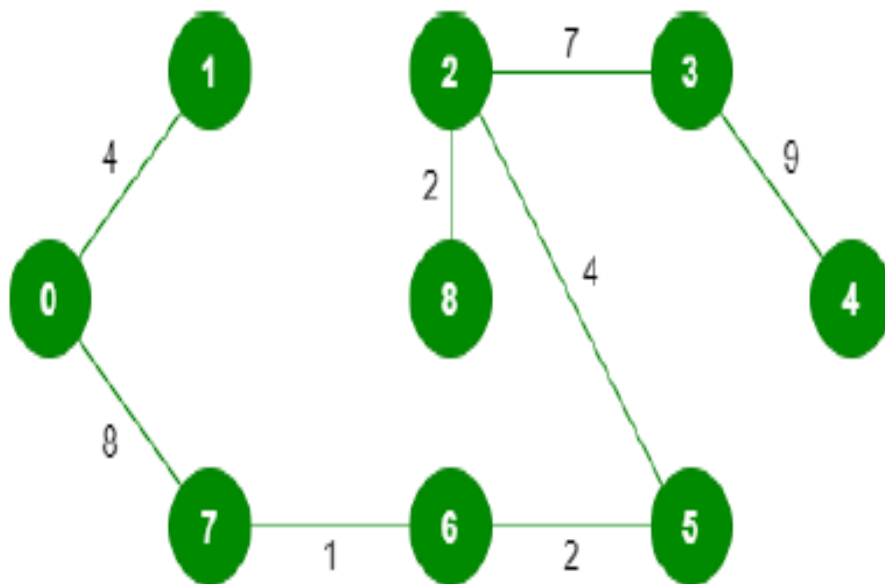
8. Pick edge 7-8: Since including this edge results in the cycle, discard it.

9. Pick edge 0-7: No cycle is formed, include it.



10. Pick edge 1-2: Since including this edge results in the cycle, discard it.

11. Pick edge 3-4: No cycle is formed, include it.



Since the number of edges included equals $(V - 1)$, the algorithm stops here.

Source Code

```
// Java program for Kruskal's algorithm to //
find Minimum Spanning Tree of a given
//connected, undirected and weighted graph
import java.util.*;
import java.lang.*;
import java.io.*;

class Graph {
    // A class to represent a graph edge
    class Edge implements Comparable<Edge>
    {
        int src, dest, weight;

        // Comparator function used for
        // sorting edgesbased on their weight public
        int compareTo(Edge compareEdge) {
            return this.weight - compareEdge.weight; }
    };

    // A class to represent a subset for
    // union-find
    class subset
    {
        int parent, rank;
    };

    int V, E; // V-> no. of vertices & E->no.of edges
    Edge edge[]; // collection of all edges

    // Creates a graph with V vertices and E edges
    Graph(int v, int e)
    {
        V = v;
        E = e;
        edge = new Edge[E];
        for (int i = 0; i < e; ++i)
            edge[i] = new Edge();
    }

    // A utility function to find set of an
```

```

// element i (uses path compression technique)
int find(subset subsets[], int i)
{
    // find root and make root as parent of i //
    (path compression)
    if (subsets[i].parent != i)
        subsets[i].parent
        = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}

// A function that does union of two sets
// of x and y (uses union by rank)
void Union(subset subsets[], int x, int y)
{
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);

    // Attach smaller rank tree under root
    // of high rank tree (Union by Rank)
    if (subsets[xroot].rank
        < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank
             > subsets[yroot].rank)
        subsets[yroot].parent = xroot;

    // If ranks are same, then make one as
    // root and increment its rank by one
    else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}

// The main function to construct MST using Kruskal's
// algorithm
void KruskalMST()
{
    // This will store the resultant MST
    Edge result[] = new Edge[V];

    // An index variable, used for result[]
    int e = 0;

```

```

// An index variable, used for sorted edges  int
i = 0;
for (i = 0; i < V; ++i)
    result[i] = new Edge();

// Step 1: Sort all the edges in non-decreasing // order
// of their weight. If we are not allowed to // change the
// given graph, we can create a copy of // array of edges
Arrays.sort(edge);

// Allocate memory for creating V subsets
subset subsets[] = new subset[V];
for (i = 0; i < V; ++i)
    subsets[i] = new subset();

// Create V subsets with single elements  for
(int v = 0; v < V; ++v)
{
    subsets[v].parent = v;
    subsets[v].rank = 0;
}

i = 0; // Index used to pick next edge

// Number of edges to be taken is equal to V-1
while (e < V - 1)
{
    // Step 2: Pick the smallest edge. And increment // the
    // index for next iteration
    Edge next_edge = edge[i++];

    int x = find(subsets, next_edge.src); int y =
    find(subsets, next_edge.dest);

    // If including this edge doesn't cause cycle, //
    // include it in result and increment the index // of
    // result for next edge
    if (x != y) {
        result[e++] = next_edge;
        Union(subsets, x, y);
    }
    // Else discard the next_edge
}

// print the contents of result[] to display

```

```

// the built MST
System.out.println("Following are the edges in " +
"the constructed MST"); int minimumCost = 0;
for (i = 0; i < e; ++i)
{
System.out.println(result[i].src + " -- " +
result[i].dest
+ " == " + result[i].weight); minimumCost
+= result[i].weight;
}
System.out.println("Minimum Cost Spanning Tree " +
minimumCost);
}

```

// Driver Code

```

public static void main(String[] args)
{

```

/* Let us create following weighted graph 10

0-----1

| \ |

6| 5 \ |15

| \ |

2-----3

4 */

int V = 4; // Number of vertices in graph int

E = 5; // Number of edges in graph

Graph graph = new Graph(V, E);

// add edge 0-1

graph.edge[0].src = 0;

graph.edge[0].dest = 1;

graph.edge[0].weight = 10;

// add edge 0-2

graph.edge[1].src = 0;

graph.edge[1].dest = 2;

graph.edge[1].weight = 6;

// add edge 0-3

graph.edge[2].src = 0;

graph.edge[2].dest = 3;

graph.edge[2].weight = 5;

```
// add edge 1-3
graph.edge[3].src = 1;
graph.edge[3].dest = 3;
graph.edge[3].weight = 15;

// add edge 2-3
graph.edge[4].src = 2;
graph.edge[4].dest = 3;
graph.edge[4].weight = 4;

// Function call
graph.KruskalMST();
}
}
// This code is contributed by Aakash Hasija
```

Output

Following are the edges in the constructed
MST 2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19

Experiment No: 6

Name of the Student: _____

Class: TE Batch: _____

Date of Performance: _____

Signature of the Subject Incharge: _____

Assignment No. 6

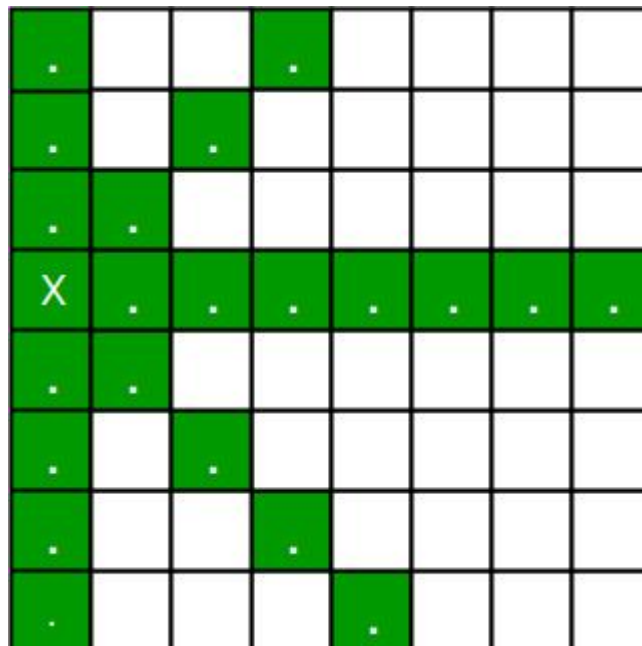
Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

N Queen Problem using Branch and Bound

The **N queen's puzzle** is the problem of placing N [chess queens](#) on an $N \times N$ chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

Backtracking Algorithm for N-Queen is already discussed [here](#). In backtracking solution we backtrack when we hit a dead end. ***In Branch and Bound solution, after building a partial solution, we figure out that there is no point going any deeper as we are going to hit a dead end.***

Let's begin by describing backtracking solution. "The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false."



1. For the 1st Queen, there are total 8 possibilities as we can place 1st Queen in any row of first column. Let's place Queen 1 on row 3.
2. After placing 1st Queen, there are 7 possibilities left for the 2nd Queen. But wait, we don't really have 7 possibilities. We cannot place Queen 2 on rows 2, 3 or 4 as those cells are under attack from Queen 1. So, Queen 2 has only $8 - 3 = 5$ valid positions left.
3. After picking a position for Queen 2, Queen 3 has even fewer options as most of the cells in its column are under attack from the first 2 Queens.

We need to figure out an efficient way of keeping track of which cells are under attack. In previous solution we kept an 8-by-8 Boolean matrix and update it each time we placed a queen, but that required linear time to update as we need to check for safe cells.

Basically, we have to ensure 4 things:

1. No two queens share a column.
2. No two queens share a row.
3. No two queens share a top-right to left-bottom diagonal.
4. No two queens share a top-left to bottom-right diagonal.

Number 1 is automatic because of the way we store the solution. For number 2, 3 and 4, we can perform updates in $O(1)$ time. The idea is to keep **three Boolean arrays that tell us which rows and which diagonals are occupied**.

Lets do some pre-processing first. Let's create two $N \times N$ matrix one for / diagonal and other one for \ diagonal. Let's call them slashCode and backslashCode respectively. The trick is to fill them in such a way that two queens sharing a same /diagonal will have the same value in matrix slashCode, and if they share same \diagonal, they will have the same value in backslashCode matrix.

For an $N \times N$ matrix, fill slashCode and backslashCode matrix using below formula –

$\text{slashCode}[\text{row}][\text{col}] = \text{row} + \text{col}$

$\text{backslashCode}[\text{row}][\text{col}] = \text{row} - \text{col} + (N-1)$

Using above formula will result in below matrices

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

$$r - c + 7$$

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

$$r + c$$

The 'N - 1' in the backslash code is there to ensure that the codes are never negative because we will be using the codes as indices in an array.

Now before we place queen i on row j, we first check whether row j is used (use an array to store row info). Then we check whether slash code (j + i) or backslash code (j - i + 7) are used (keep two arrays that will tell us which diagonals are occupied). If yes, then we have to try a

SRTTC FOE LABORATORY PRACTICE II 37

different location for queen i. If not, then we mark the row and the two diagonals as used and recurse on queen i + 1. After the recursive call returns and before we try another position for queen i, we need to reset the row, slash code and backslash code as unused again, like in the code from the previous notes.

Below is the implementation of above idea –

```
""" Python3 program to solve N Queen Problem
```

```
using Branch or Bound """
```

```
N = 8
```

```
""" 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]):

```

SRTTC FOE LABORATORY PRACTICE II 38

```

    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 """
```

SRTTC FOE LABORATORY PRACTICE II 39

```
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

SRTTC FOE LABORATORY PRACTICE II 40

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
```

SRTTC FOE LABORATORY PRACTICE II 41

```
            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()

```

Output :

```

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

```

SRTTC FOE LABORATORY PRACTICE II 42

Performance:

When run on local machines for $N = 32$, the backtracking solution took 659.68 seconds while above branch and bound solution took only 119.63 seconds. The difference will be even huge for larger values of N .

Backtracking Algorithm

```
Process returned 0 (0x0)   execution time : 659.686 s
Press any key to continue.
```

Branch and Bound Algorithm

```
Process returned 0 (0x0)   execution time : 119.631 s
Press any key to continue.
```


Experiment No: 7

Name of the Student: _____

Class: TE Batch: _____

Date of Performance: _____

Signature of the Subject Incharge: _____

Assignment No. 7

Develop an elementary chatbot for any suitable customer interaction application.

Introduction

Chatbot is a python based project. A chatbot is a computer program that interacts with human conversation through voice or text. It is built using python programming.

The chatbot project is using python version 3. It is an interesting project. Python is a general purpose high-level programming. It is useful for developing desktop GUI, websites, and web apps. This is a command-line based project. This project chatbot is a beginner or school project. It is coded in a simple way that every learner will understand.

When you execute the project you will see a set of questions in a command line. The chatbot will ask a set of questions and you have to answer theme. You must start the conversation answering the question asked by the boy. It is an entertainment-based project. In this project you must import time and random. The bot will mostly use random input. Time is useful for adding and waiting for the user's response.

DOWNLOAD PROJECT:

<https://code-mentor.org/chatbot-in-python-with-source-code/>

PROCEDURE:

How to use this project?

- Install python version 3.
- Download the project.
- Extract the zip file and get the code.
- Set up an editor or IDE. (vs code, pycharm, anaconda)
- Open the python file in an editor.
- Execute the program.
- Start conversation with bot.
- Enjoy and Share!

Output

```

Hello, what is your name? code
Hello code
How are you today? good
I'm feeling good too!
What is your favourite colour? blue
My favourite colour is Blue
What are you doing today? hiking
I feel like watching TV today
Do you have a girlfriend? yes
cool
Which food type do you like most? asian
I mostly like Japanese
Which sport do you love to watch the most? soccer
I love to watch Baseball
I love talking with you!
Do you know the height of world's tallest mountain? 8848
8848
Anyone can answer that.
How many planet are there in this solar system 34
I guess there are 8 planets
What is the name of your horscope? leo
My horscope name is Pisces

```

The image above is the screenshot of the output. This is how you must chat with bot. It is a simple beginner project. There is no necessary to import all the complex packages. Such as [nltk](#), [sklearn](#), [tkinter](#) and many others. This project will be fruitful for beginners to enhance their skills. You will learn new skills in this code. The project will be productive for you. Download the project and use the code by yourself.

Benefits of Bots –

1. Understandable information about the customer.
2. Can be called a selling partner by making and sending the products information.
3. Provides 24hrs services
4. Satisfy the need of clients as the customer will not go on waiting for your call. They need the action quickly or will turn to another brand.
5. Most of the customer prefers sending messages, text, SMS to the company for information. Marketing Bot can result or give your Business growth by making higher sales and satisfying the needs. Facebook Messenger is one of the widely used messengers in the U.S.
6. Recently chatbots were used by World Health Organization for providing information by ChatBot on Whatsapp.
7. Facebook Messenger, Slack, Whatsapp, and Telegram make use of ChatBot.
8. The modern need is there for Bot Building for growth of Business to make progress.
9. Another example of making use of ChatBo is Google Assistant and Siri.
10. Bots, for the most part, operate on a network. Bots that can communicate with one another will

use internet-based services like IRC.

SRTTC FOE LABORATORY PRACTICE II 46

Conclusion

This article is the base of knowledge of the definition of ChatBot, its importance in the Business, and how we can build a simple Chatbot by using Python and Library Chatterbot.

Experiment No: 8

Name of the Student: _____

Class: TE Batch: _____

Date of Performance: _____

Signature of the Subject Incharge: _____

Assignment No. 8

Implement any one of the following Expert System

I. Information management

What is Expert System in AI (Artificial Intelligence)? With Example

What is Expert System?

Expert System is an interactive and reliable computer-based decision-making system which uses both facts and heuristics to solve complex decision-making problems. It is considered at the highest level of human intelligence and expertise. The purpose of an expert system is to solve the most complex issues in a specific domain.

Expert Systems in Artificial Intelligence

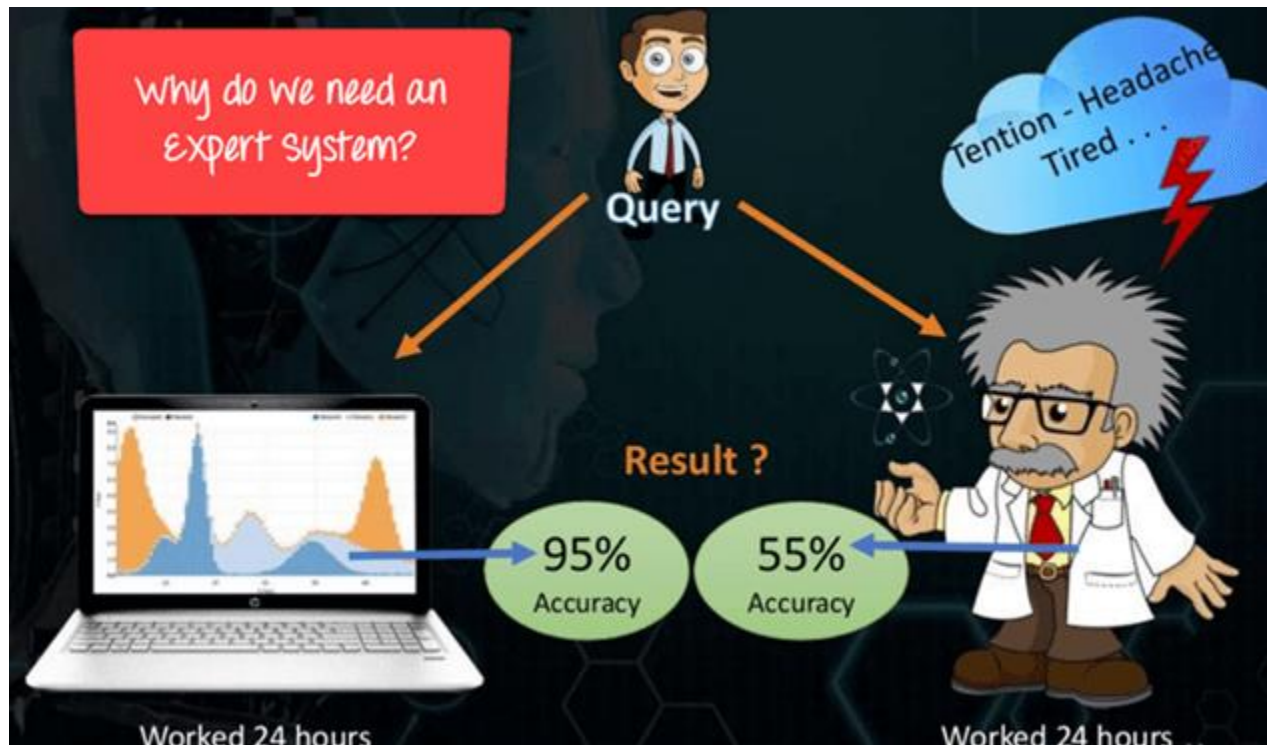
The Expert System in AI can resolve many issues which generally would require a human expert. It is based on knowledge acquired from an expert. Artificial Intelligence and Expert Systems are capable of expressing and reasoning about some domain of knowledge. Expert systems were the predecessor of the current day artificial intelligence, deep learning and machine learning systems.

Examples of Expert Systems

Following are the Expert System Examples:

- **MYCIN:** It was based on backward chaining and could identify various bacteria that could cause acute infections. It could also recommend drugs based on the patient's weight. It is one of the best Expert System Example.
- **DENDRAL:** Expert system used for chemical analysis to predict molecular structure.
- **PXDES:** An Example of Expert System used to predict the degree and type of lung cancer
- **CaDet:** One of the best Expert System Example that can identify cancer at early stages

Characteristics of Expert System

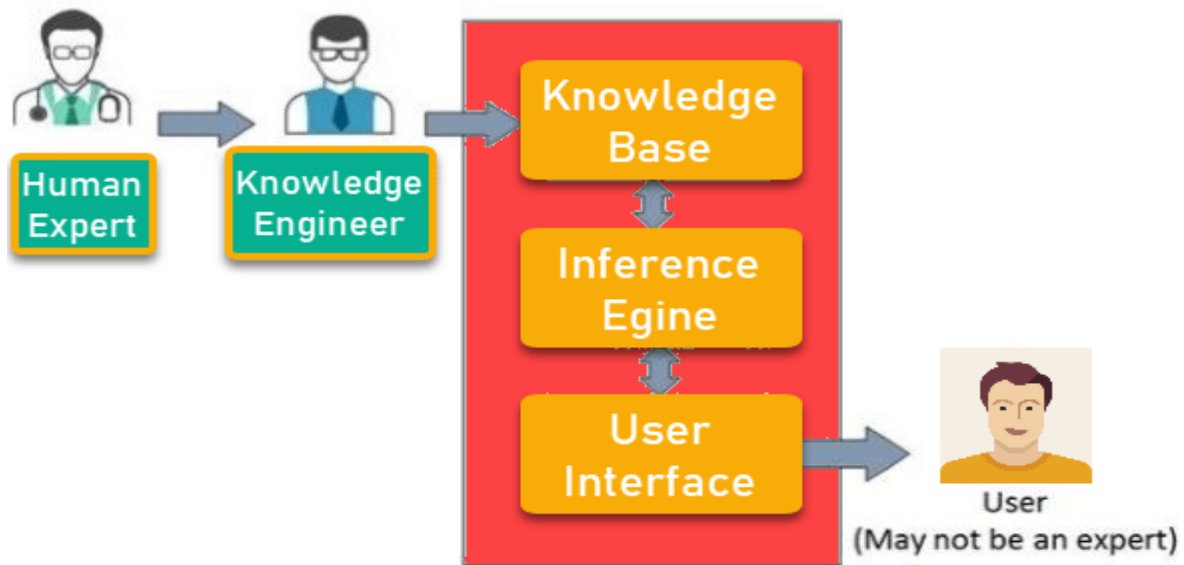


Why Expert Systems are required?

Following are the important Characteristics of Expert System in AI:

- **The Highest Level of Expertise:** The Expert system in AI offers the highest level of expertise. It provides efficiency, accuracy and imaginative problem-solving.
- **Right on Time Reaction:** An Expert System in Artificial Intelligence interacts in a very reasonable period of time with the user. The total time must be less than the time taken by an expert to get the most accurate solution for the same problem.
- **Good Reliability:** The Expert system in AI needs to be reliable, and it must not make any a mistake.
- **Flexible:** It is vital that it remains flexible as it the is possessed by an Expert system.
- **Effective Mechanism:** Expert System in Artificial Intelligence must have an efficient mechanism to administer the compilation of the existing knowledge in it.
- **Capable of handling challenging decision & problems:** An expert system is capable of handling challenging decision problems and delivering solutions.

Components of Expert System



The Expert System in AI consists of the following given components:

User Interface

The user interface is the most crucial part of the Expert System Software. This component takes the user's query in a readable form and passes it to the inference engine. After that, it displays the results to the user. In other words, it's an interface that helps the user communicate with the expert system.

Inference Engine

The inference engine is the brain of the expert system. Inference engine contains rules to solve a specific problem. It refers the knowledge from the Knowledge Base. It selects facts and rules to apply when trying to answer the user's query. It provides reasoning about the information in the knowledge base. It also helps in deducing the problem to find the solution. This component is also helpful for formulating conclusions.

Knowledge Base

The knowledge base is a repository of facts. It stores all the knowledge about the problem domain. It is like a large container of knowledge which is obtained from different experts of a specific field.

Thus we can say that the success of the Expert System Software mainly depends on the highly accurate and precise knowledge.

Other Key terms used in Expert Systems

Facts and Rules

A fact is a small portion of important information. Facts on their own are of very limited use. The rules are essential to select and apply facts to a user problem.

Knowledge Acquisition

The term knowledge acquisition means how to get required domain knowledge by the expert system. The entire process starts by extracting knowledge from a human expert, converting the acquired knowledge into rules and injecting the developed rules into the knowledge base.



Knowledge Extraction Process

Participant in Expert Systems Development

Participant Role

Domain Expert	He is a person or group whose expertise and knowledge is taken to develop an expert system
Knowledge Engineer	Knowledge engineer is a technical person who integrates knowledge into computer systems
End User	It is a person or group of people who are using the expert system to get advice which will not be provided by the expert

The process of Building An Expert Systems

Conventional System	Expert System
Knowledge and processing are combined in one unit.	Knowledge database and the processing mechanism are two separate components.
Human Expert	Artificial Expertise
Perishable	Permanent
Difficult to Transfer	Transferable
Difficult to Document	Easy to Document
Unpredictable	Consistent
Expensive	Cost effective System

- Determining the characteristics of the problem
 - Knowledge engineer and domain expert work in coherence to define the problem ·
- The knowledge engineer translates the knowledge into a computer-understandable language. He designs an inference engine, a reasoning structure, which can use knowledge when needed.
- Knowledge Expert also determines how to integrate the use of uncertain knowledge in the reasoning process and what type of explanation would be useful.

Conventional System vs. Expert System

The programme does not make errors (Unless error in programming).	The Expert System may make a mistake.
The system is operational only when fully developed.	The expert system is optimized on an ongoing basis and can be launched with a small number of rules.
Step by step execution according to fixed algorithms is required.	Execution is done logically & heuristically.
It needs full information.	It can be functional with sufficient or insufficient information.

Human expert vs. Expert System

Advantages of Expert System

Below are the main advantages/benefits of Expert Systems in Artificial Intelligence (AI):

- It improves the decision quality
- Cuts the expense of consulting experts for problem-solving
- It provides fast and efficient solutions to problems in a narrow area of specialization. ·
- It can gather scarce expertise and use it efficiently.
- Offers consistent answer for the repetitive problem
- Maintains a significant level of information
- Helps you to get fast and accurate answers
- A proper explanation of decision making
- Ability to solve complex and challenging issues
- Artificial Intelligence Expert Systems can steadily work without getting emotional, tensed or fatigued.

Limitations of Expert System

Below are the disadvantages/limitations of Expert System in AI:

- Unable to make a creative response in an extraordinary situation
- Errors in the knowledge base can lead to wrong decision
- The maintenance cost of an expert system is too expensive
- Each problem is different therefore the solution from a human expert can also be different and more creative

Applications of Expert Systems

Some popular Application of Expert System:

- Information management
- Hospitals and medical facilities
- Help desks management

- Employee performance evaluation
- Loan analysis
- Virus detection
- Useful for repair and maintenance projects
- Warehouse optimization
- Planning and scheduling
- The configuration of manufactured objects
- Financial decision making Knowledge publishing
- Process monitoring and control
- Supervise the operation of the plant and controller
- Stock market trading
- Airline scheduling & cargo schedules

Summary

· An Expert System is an interactive and reliable computer-based decision-making system which uses both facts and heuristics to solve complex decision-making problem · Key components of an Expert System are

1) User Interface, 2) Inference Engine, 3) Knowledge Base

· Key participants in Artificial Intelligence Expert Systems Development are

1) Domain Expert 2) Knowledge Engineer 3) End User

· Improved decision quality, reduce cost, consistency, reliability, speed are key benefits of an Expert System

· An Expert system can not give creative solutions and can be costly to maintain. · An Expert System can be used for broad applications like Stock Market, Warehouse, HR, etc

If you want to learn about Artificial Intelligence, here's a free tutorial you'll want to check out: [AI Tutorial](#)

Student Information Management System In C++ With Source Code

Project: Student Information System

Student Information System is based on the concept to generate the Student's Information records and to add & update it. Here User can add their Student's detail with their courses safely and it's not time-consuming. This System makes easy to store records of each and every employee. The whole project is designed in 'C++' language and different variables and strings have been used for the development of this project. This mini project is easy to operate and understand by the users.

Features:

1. Proper Log-In System.
2. Easy To Add, Modify, List And Delete Student's Information Records.
3. Password Protected.

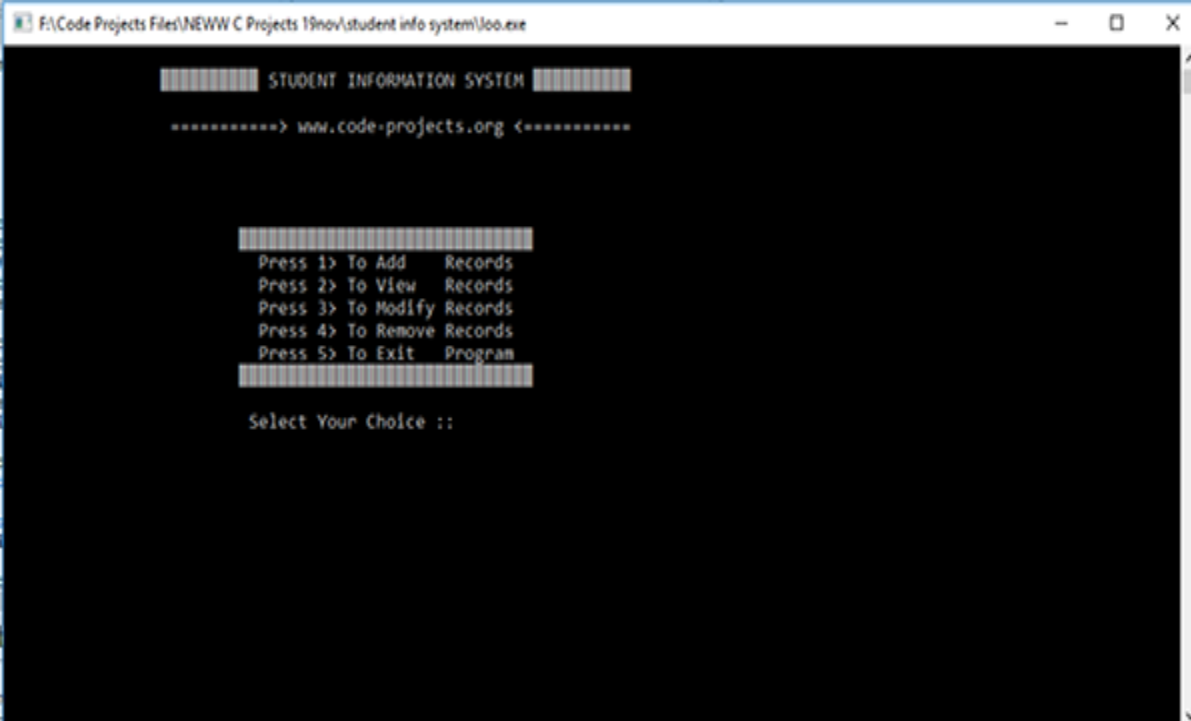
DOWNLOAD STUDENT INFORMATION SYSTEM WITH SOURCE CODE :

CLICK THE BUTTON BELOW

DOWNLOAD PROJECT:

<https://download.code-projects.org/details/a98005e8-1910-4dd5-b893-f925404d99e8>

OUTPUT:



```
F:\Code Projects Files\NEWW C Projects 19nov\student info system\joo.exe
===== STUDENT INFORMATION SYSTEM =====
*****> www.code-projects.org <*****

Press 1> To Add    Records
Press 2> To View  Records
Press 3> To Modify Records
Press 4> To Remove Records
Press 5> To Exit  Program

Select Your Choice ::
```