**Project Title:**

**Understanding and Implementing Tree Data Structures**

Submitted by:

AARUSH C S

ATHUL K KOSHY

MOHAMMED RIZWAN PP

MOIDEEN NIHAL

Course: BCA (Hons) AI

Subject: Data Structures and Algorithms

**Objective:**

To implement a **Binary Tree** and understand the fundamental operations such as **insertion**, **traversals (inorder, preorder, postorder)**, and **searching**, while exploring the practical applications of trees in computing.

**What is a Tree?**

A **tree** is a hierarchical data structure consisting of nodes. The top node is called the **root**, and every node can have child nodes. It is widely used in real-life applications such as:

* File systems
* Hierarchical databases
* Expression parsing
* AI (decision trees)

**Types of Trees (Brief Overview):**

* **Binary Tree** – Each node has at most two children.
* **Binary Search Tree (BST)** – Binary tree with sorted node placement.
* **AVL Tree** – Self-balancing BST.
* **Heap** – Tree-based data structure used in priority queues.
* **Trie** – Tree used for efficient searching of words (prefix trees).

For this project, we will focus on the **Binary Tree**.

**Tree Operations Covered:**

1. **Insert Node**
2. **Inorder Traversal**
3. **Preorder Traversal**
4. **Postorder Traversal**
5. **Search Node (Optional)**

**Code Implementation (Python):**

**Binary Tree with Traversals**

python

CopyEdit

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.right = None

# Insertion (Level Order for simplicity)

def insert\_level\_order(arr, root, i, n):

if i < n:

temp = Node(arr[i])

root = temp

root.left = insert\_level\_order(arr, root.left, 2 \* i + 1, n)

root.right = insert\_level\_order(arr, root.right, 2 \* i + 2, n)

return root

# Traversals

def inorder(root):

if root:

inorder(root.left)

print(root.data, end=' ')

inorder(root.right)

def preorder(root):

if root:

print(root.data, end=' ')

preorder(root.left)

preorder(root.right)

def postorder(root):

if root:

postorder(root.left)

postorder(root.right)

print(root.data, end=' ')