Second Project

Identifying and Describing Data Structures:

Data structures are essential components of programming, enabling the organization and efficient access to data. Here's a comprehensive list of commonly used data structures:

1. Arrays:
- ❖ Arrays are collections of elements stored in contiguous memory locations, allowing for random access to elements using indices.
- ❖ They have constant-time access but linear-time complexity for insertion and deletion.

2. Linked Lists:
- ❖ Linked lists consist of nodes, where each node contains data and a reference to the next node in the sequence.
- ❖ They support constant-time insertion and deletion but require linear-time traversal for access.

3. Stacks:
- ❖ Stacks follow the Last In, First Out (LIFO) principle, allowing elements to be inserted and removed from the same end (the top).
- ❖ They support constant-time insertion and deletion.

4. Queues:
- ❖ Queues follow the First In, First Out (FIFO) principle, allowing elements to be inserted at the rear and removed from the front.
- ❖ They support constant-time insertion and deletion.

5. Binary Trees:
- ❖ Binary trees are hierarchical data structures consisting of nodes, where each node has at most two children: left and right.
- ❖ They support efficient search, insertion, and deletion operations with logarithmic time complexity.

6. Graphs:
- ❖ Graphs are non-linear data structures consisting of vertices (nodes) and edges (connections between vertices).

- ❖ They can be represented using adjacency lists, adjacency matrices, or other methods and support various operations like traversal and pathfinding.

7. Hash Tables:

- ❖ Hash tables store key-value pairs and use a hash function to compute an index for efficient data retrieval.
- ❖ They provide constant-time average-case complexity for insertion, deletion, and retrieval operations.

Development in PHP:

Below are basic implementations of the identified data structures in PHP:

```php
// Array implementation
class MyArray {
    private $array;
    private $size;

    public function __construct($capacity) {
        $this->array = array_fill(0, $capacity, null);
        $this->size = 0;
    }

    // Implement methods for insertion, deletion, access, etc.
}

// Linked list implementation
class ListNode {
    public $val;
    public $next;

    public function __construct($val) {
        $this->val = $val;
        $this->next = null;
    }
}

class MyLinkedList {
    private $head;
```

```
    // Implement methods for insertion, deletion, traversal, etc.
}

// Stack implementation
class MyStack {
    private $array;
    private $top;

    // Implement methods for push, pop, peek, etc.
}

// Queue implementation
class MyQueue {
    private $array;
    private $front;
    private $rear;

    // Implement methods for enqueue, dequeue, peek, etc.
}

// Binary tree implementation
class TreeNode {
    public $val;
    public $left;
    public $right;

    public function __construct($val) {
        $this->val = $val;
        $this->left = null;
        $this->right = null;
    }
}

class BinaryTree {
    private $root;

    // Implement methods for insertion, deletion, traversal, etc.
}
```

```
// Graph implementation (using adjacency list)
class Graph {
    private $V;
    private $adjList;

    public function __construct($V) {
        $this->V = $V;
        $this->adjList = array_fill(0, $V, []);
    }

    // Implement methods for adding edges, traversal, etc.
}

// Hash table implementation
class MyHashTable {
    private $capacity;
    private $table;

    public function __construct($capacity) {
        $this->capacity = $capacity;
        $this->table = array_fill(0, $capacity, null);
    }

    // Implement methods for insertion, deletion, searching, etc.
}
```

Conclusion:

In conclusion, data structures are fundamental components of programming, enabling efficient organization and access to data. By implementing these data structures in PHP, we deepen our understanding of their principles and enhance our ability to design robust and efficient software solutions, regardless of the programming paradigm being employed.