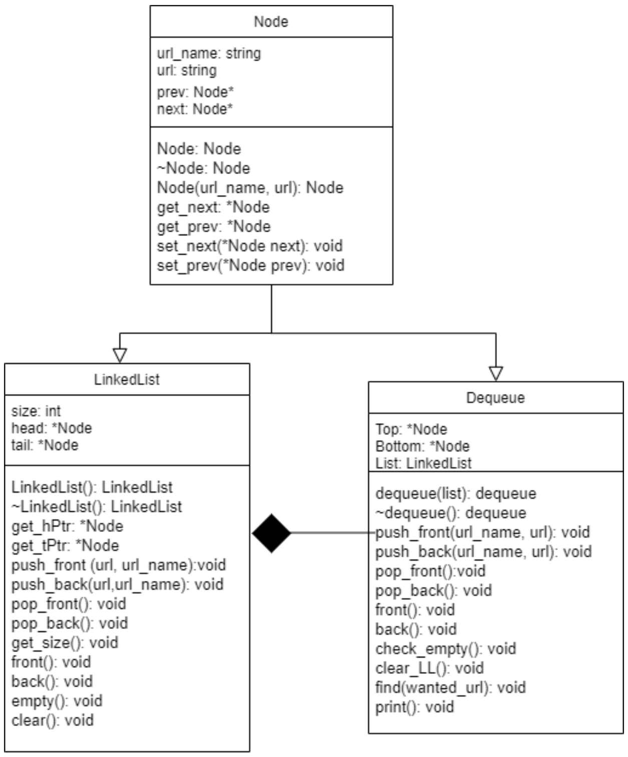
1. **Overview of Classes**

   There are tree classes in my project design: Node, LinkedList, and Dequeue.
   - The Node class stores the data of the url, the url name and pointers to the next and previous node.
   - The LinkedList class connects a collection of nodes and contains functions to make changes to the list.
   - The dequeue invokes the linked list class and stores the items within the linked list in reverse order to be used for specific implementations that require the list to be traversed in reverse order.
2. **ULM Class Design**



3. **Details on Design Decisions**

Node Class:

- Chose to make Node a class rather than struct and made attributes private to invoke the use of setter and getter functions to make changes or access these variables within other classes.
- Node stores previous and next pointer to allow for the implementation of doubly linked list.

Linked List Class:

- Stores the head, tail and size as private member variables, constructor initializes pointers to null and sets size to zero.

- Implemented getters for head and tail pointers as a way for the dequeue class to have access to them.
- Push_front adds node to front of list at head, push_back adds node to back of list at the tail.
- Pop front or back removes node at tail or head of the list.
- Get_size function used to allow dequeue class to access size of list.
- Front and back return the node at head and tail respectively.
- Clear function loops through list using while loop and deallocates memory at each node.

Dequeue Class:

- Acts as a wrapper or interface for linked list class, top and bottom pointers point to head and tail of linked list associated with dequeue object.
- All functions within dequeue that are repeats of the functions of the linked class are used to call the linked list function on the linked list object in the dequeue and then update private variables of the dequeue if required.
- Find searches list from the top to bottom using a while loop to compare the url name at each node with the input string.
- Print uses the top pointer and feature of doubly linked list to print from top to bottom (tail to head), implemented using while loop.

4. **Test Cases**
   - Added additional test cases to implement the basic functionality and specifications of the project
   - Tested input string manipulation to ensure correct url name and urls were extracted correctly, for example if name of url contains spacing
   - Tested logical handling of running commands if list is empty
   - All nodes are correctly removed after calling clear
   - Tested logic for popping to ensure that it is successful when there are still elements in the linked list
   - Tested find functionality to ensure it could run correctly with spaces and special characters
   - Tested different formats for inputting new nodes into the list with different formats
5. **Performance Considerations**

   To ensure run time requirements are met I implemented a double linked list to ensure any linked list manipulation done at the back could be achieved with a constant run time. All functions implemented have a constant run time with the exception of print, find, and clear.