API CreateTopic (node, topic)

The Create Topic API is used to create a topic by calling the client in the specified peer node. After calling the Create Topic API, we first check if the given topic is equal to "exit." If it is, the user can quit that option and return to the main menu of APIs. The server first checks if such a topic exists in the Message Buffer. If the topic is not present, the server adds (node, topic) as a set to the dictionary. The server retrieves the credentials of the node specified in the input arguments from its directory and provides them to the calling client. The calling client will use these credentials to send the topic name to that node. The responsive node will receive the request and add the topic to its buffer. Since the topic has just been created, there should be no subscribers to that topic; thus, the subscription and message buffer dictionaries will be empty. After completion, we can return an acknowledgment indicating whether the topic has been added.

API DeleteTopic (topic)

Similar to the Create Topic API, this API deletes a topic by calling the client. The request is sent to the server first. If the topic exists in the server's topics dictionary, it removes that topic from its own buffer. The server also identifies the holder of the topic and provides those credentials to the calling peer node. The peer node then sends that request to the topic holder node. The topic holder node receives the request and deserializes it. It first checks if the calling peer node is subscribed to the topic. If so, it removes the topic from its directory and deletes it from the subscriptions dictionary.

API Send (topic, message)

The peer node sends the message to the specified topic. The request reaches the index server, which finds the owner of the topic. It returns the IP and port credentials of the topic holder to the calling peer node. The calling peer node then forwards this information to the topic holder node. The topic holder deserializes the incoming request and fetches the message. After that, it checks if the topic exists in the message buffer. If the topic is in the buffer, it appends the message to messageBuffer[topic]. We also check for empty messages to ensure that we do not accept them.

API Subscribe (topic)

When a client requests to subscribe to a topic, we check if that topic exists in the system. The server receives the request and deserializes it. It fetches the topic and looks up the topic owner in its directory. Afterward, it passes the credentials to the caller for forwarding. The caller then sends that request to the corresponding peer node. The callee receives the request and deserializes it. It then adds the caller peer node to the list of subscribers for the given topic. Finally, it returns an acknowledgment indicating that the caller has been subscribed to the topic. If the topic does not exist, the server responds to the caller that such a topic does not exist. We also check if the client is already subscribed. After this, we add the client to the subscription dictionary and initialize the reading index to 0. Since the client has just subscribed, they have not read any messages, so we need to initialize it to 0.

API Pull (topic)

First, the server checks if the topic exists. If the topic exists, it looks up the topic holder and returns the connection credentials of the topic holder to the calling peer node. The caller then sends the request to the topic owner. The topic owner receives the request and deserializes it. It first checks if the calling peer node is subscribed to the topic. If not, it returns a message indicating that the caller is not subscribed. Otherwise, it retrieves the index of the last read message from the read history and finds the unread messages for the calling peer node. If there are new messages, it returns the list of messages. If there

are no new messages for the calling peer node, it simply sends back "Empty." After each pull, we call the garbage collector to check if all subscribers have read the messages from that topic. If so, all old messages are removed from the message buffer, and the indexes of the subscribers are reset to 0.

Unregister(peerName):
This API will unregister the peer node from the index server. When we choose this option, we are prompted to give two options:

1. Delete all existing topics and unregister the peer node. This will erase all topics from the list and from the index server.
2. Transfer topics and unregister. This will transfer all topics, subscriptions of the topics, and the read history to the specified node. The server index will also update its records as the ownership has changed. Afterward, the server index removes the peer node from the list.