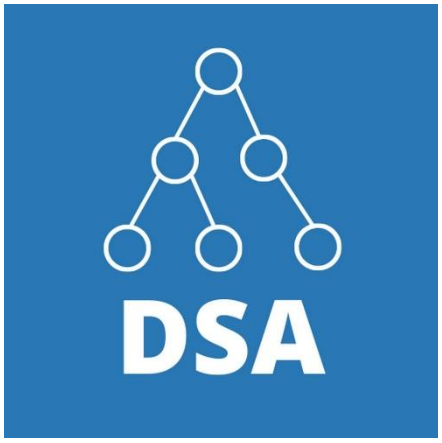
AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

19CSE212- DSA CAPSTONE REPORT



Chat List Using LRU Cache

1) Team Members:

S. No	Name	Roll no:
1.	Aman Kshetri	CH.EN.U4CSE21003
2.	Raj Sah Rauniyar	CH.EN.U4CSE21051
3.	Saroj Kumar Bhagat	CH.EN.U4CSE21060
4.	Tej Mahesh	CH.EN.U4CSE21041

2) Abstract:

We have implemented a Chat List web application using the LRU (Least Recently Used) Cache.

It provides insertion, deletion, and searching in constant time. When a message from a new chat arrives, it will be inserted at the top of the chat list. If the message is from an existing conversation, then it will update the chat position to the top.

LRU cache stand for Least Recently Used Cache. which evict least recently used entry. As Cache purpose is to provide fast and efficient way of retrieving data. it needs to meet certain requirement.

In case of LRU cache we evict least recently used entry so we have to keep track of recently used entries, entries which have not been used from long time and which have been used recently. plus, lookup and insertion operation should be fast enough.

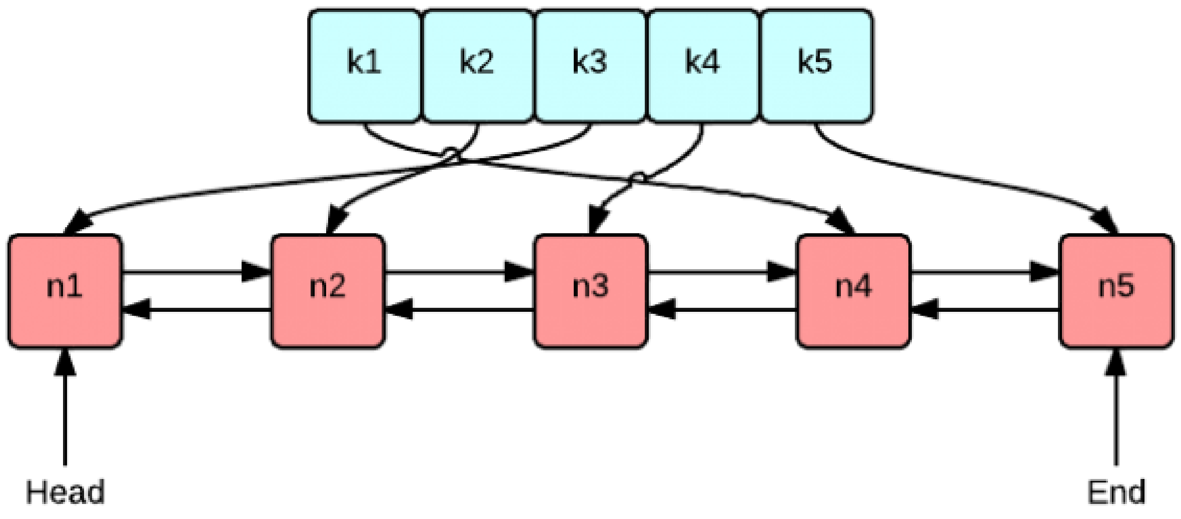
3) Data Structures used:

A LRU cache is designed by combining two data structures: a Linked List and a Hash Map.

- Linked List
- Hash Maps

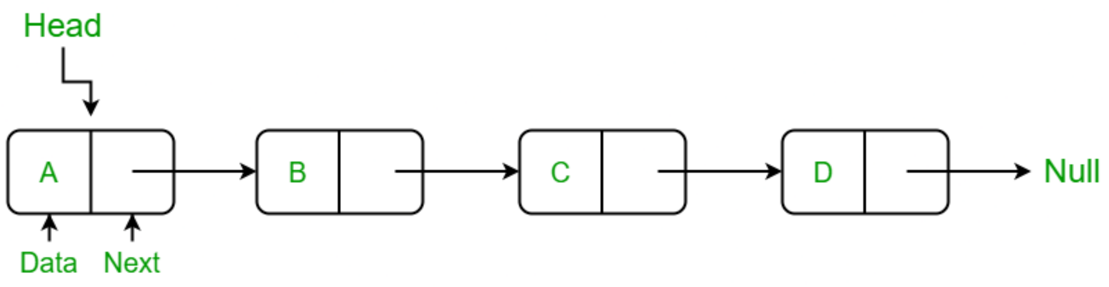
So, our Implementation of LRU cache will have HashMap and Doubly LinkedList. In Which HashMap will hold the keys and address of the Nodes of Doubly LinkedList. And Doubly LinkedList will hold the values of keys.

As we need to keep track of Recently used entries, we will use a clever approach. We will remove element from bottom and add element on start of LinkedList and whenever any entry is accessed, it will be moved to top. so that recently used entries will be on Top and Least used will be on Bottom.



a) Linked List

⇨ A linked list is a data structure that consists of a sequence of nodes, where each node contains data and a reference (or pointer) to the next node in the sequence. Unlike arrays, linked lists do not have a fixed size and can dynamically grow or shrink as elements are added or removed.

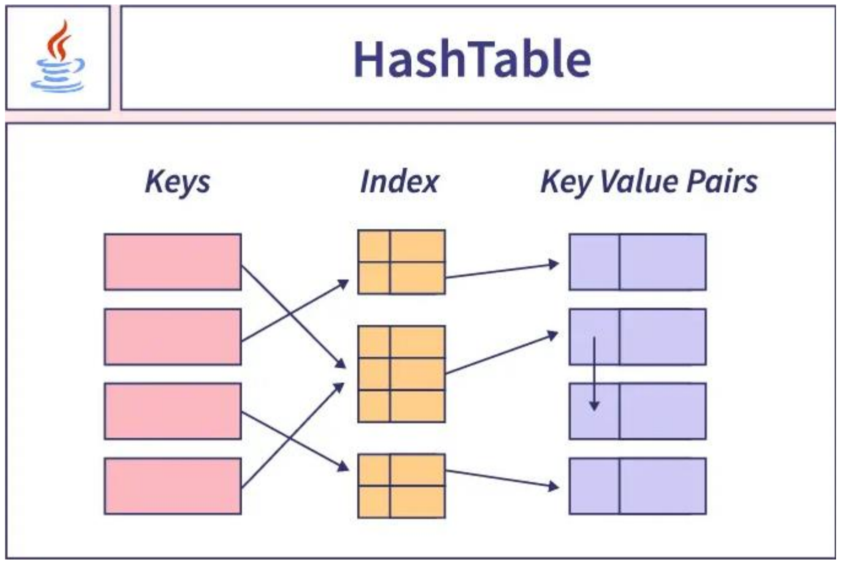


The first node in a linked list is called the head, and the last node points to null, indicating the end of the list. Each node contains data and a reference to the next node, allowing sequential access to the elements. Linked lists can be singly linked, where each node has a reference to the next node, or doubly linked, where each node has references to both the next and previous nodes, enabling bidirectional traversal.

Linked lists are efficient for inserting and deleting elements at any position, as they involve updating only a few pointers. However, accessing elements in a linked list is slower compared to arrays, as it requires traversing the list from the beginning.

b) Hash Maps

⇨ A hash map, also known as a hash table or associative array, is a data structure that enables efficient lookup, insertion, and deletion operations. It stores data in key-value pairs, where each key is unique and associated with a value. Hash maps use a hashing function to compute an index, called a hash code, based on the key. This index is used to store and retrieve the corresponding value.



Internally, a hash map typically consists of an array (or a bucket array) of fixed or dynamically resizing size. The hash code generated from the key determines the index in the array where the key-value pair is stored. If multiple keys have the same hash code (known as a collision), the hash map uses separate chaining or open addressing techniques to handle collisions.

Hash maps offer constant-time complexity O(1) for average case operations, such as insertion, deletion, and retrieval, making them highly efficient for large datasets. However, in the worst-case scenario, when there are many collisions, the time complexity may increase to O(n), where n is the number of elements in the hash map.

4) Source Code:

The Source Code for our Chat-List Application is given below: The attachment contains HTML, CSS and JS files.

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Chatlist Caching</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></
script>
  <script src="https://kit.fontawesome.com/5548f5ed00.js"
crossorigin="anonymous"></script>
  <link href="https://netdna.bootstrapcdn.com/bootstrap/4.0.0-
beta/css/bootstrap.min.css" rel="stylesheet" />
  <link rel="stylesheet" href="style.css" />
  <script src="script.js" type="module"></script>
</head>

<body>
  <nav id="navbar" class="navbar">
    Chat List Using LRU Cache
  </nav>
  <div id="container">
    <div class="container2">
      <div class="ks-messenger">
        <div class="ks-discussions">
          <div class="ks-header">
            <span class="col-4" style="text-align: center;">
              <i class="fas fa-comments" style="color: dodgerblue;"></i>
            </span>
            <span class="col-4" style="text-align: center;">
              <i class="fas fa-video" style="color: dodgerblue;"></i>
            </span>
            <span class="col-4" style="text-align: center;">
              <i class="fa fa-user" style="color: dodgerblue;"></i>
            </span>
```

```html
          </div>
          <div class="ks-body ks-scrollable jspScrollable" data-auto-
height="" style="
                height: 420px;
                overflow-y: auto;
                overflow-x: hidden;
                padding: 0px;
                width: 339px;
              " tabindex="0">
            <div class="jspContainer">
              <div class="jspPane" style="padding: 0px; top: 0px; width:
339px;">
                <ul class="ks-items" id="chat-list"></ul>
              </div>
              <div class="jspVerticalBar">
                <div class="jspCap jspCapTop"></div>
                <div class="jspTrack" style="height: 100px;">
                  <div class="jspDrag">
                    <div class="jspDragTop"></div>
                    <div class="jspDragBottom"></div>
                  </div>
                </div>
                <div class="jspCap jspCapBottom"></div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="container2" style="overflow-y: scroll; overflow-x: hidden;
padding-top: 20px;">
      <span id="temptext" style="
            height: 100%;
            width: 100%;
            text-align: center;
            font-size: x-large;
          ">
        <b>Click on generate new step to get updates!</b>
      </span>
    </div>
  </div>
  <div class="container3">
    <br />
    <button type="button" class="btn btn-danger" id="generate-step"
style="margin: auto;">
```

```
      Generate New Step
    </button>
  </div>
  <template>
    <li class="ks-item" id="ChatListItem">
      <a href="#">
        <span class="ks-avatar">
          <img src="./images/avatar1.png" width="36" height="36"
id="Image" alt="ks-avatar" />
        </span>
        <div class="ks-body">
          <div class="ks-name">
            <span id="Name">Aman Kshetri</span>
            <span id="Time" class="ks-datetime">just now</span>
          </div>
          <div class="ks-message" id="Message">
            Why didn't he come and talk to me ...
          </div>
        </div>
      </a>
    </li>
  </template>
</body>

</html>
```

Style.css

```
/*CSS Code*/

html,
body {
  margin: 0;
  padding: 0;
  height: 100%;
  /* background: #eeeeee; */
  background: #FFF4E0;
  overflow: hidden;
}

#navbar {
  font-size: 30px;
  height: 60px;
  display: flex;
```

```css
  font-family: Arial, Helvetica, sans-serif;
  justify-content: center;
  align-items: center;
  color: white;
  background-color: #008cba;
}

#container {
  width: 100%;
  height: 70%;
  display: flex;
  margin: 0 auto;
}

.container2 {
  width: 50%;
  height: 100%;
  border: 1px solid lightgray;
  display: flex;
  flex-wrap: wrap;
  align-content: center;
}

.container3 {
  padding: 50px;
  display: flex;
  flex-wrap: wrap;
  align-content: center;
}

.ks-messenger {
  margin: auto;
  margin-top: 60px;
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: inline-block;
  height: 100%;
}

.ks-messenger .ks-discussions {
  border-radius: 5px 5px 5px 5px;
  background: #fff;
  width: 340px;
  border-right: 1px solid #dee0e1;
```

```css
}

.ks-messenger .ks-discussions>.ks-header {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-align: center;
  -webkit-align-items: center;
  -ms-flex-align: center;
  align-items: center;
  height: 60px;
  border-bottom: 1px solid #dee0e1;
}

.ks-messenger .ks-discussions>.ks-search .form-control {
  border: none;
  padding: 28px 20px;
}

.ks-messenger .ks-discussions>.ks-search .icon-addon {
  color: rgba(58, 82, 155, 0.6);
}

.ks-messenger .ks-discussions>.ks-body .ks-items {
  list-style: none;
  padding: 0;
  margin: 0;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item {
  border-bottom: 1px solid #dee0e1;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  -js-display: flex;
  display: flex;
  -webkit-box-align: center;
  -webkit-align-items: center;
  -ms-flex-align: center;
  align-items: center;
  color: #333;
  padding: 15px 20px;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-group-
amount {
  position: relative;
  top: 3px;
  margin-right: 12px;
  width: 36px;
  height: 36px;
  background-color: rgba(57, 81, 155, 0.1);
  text-align: center;
  line-height: 36px;
  -webkit-border-radius: 50%;
  border-radius: 50%;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-group-
amount>.ks-badge {
  position: absolute;
  bottom: -1px;
  right: -3px;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-avatar {
  position: relative;
  top: 3px;
  margin-right: 12px;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-avatar>img
{
  width: 36px;
  height: 36px;
  -webkit-border-radius: 50%;
  border-radius: 50%;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-
avatar>.ks-badge {
  position: absolute;
  bottom: -3px;
  right: -3px;
}
```

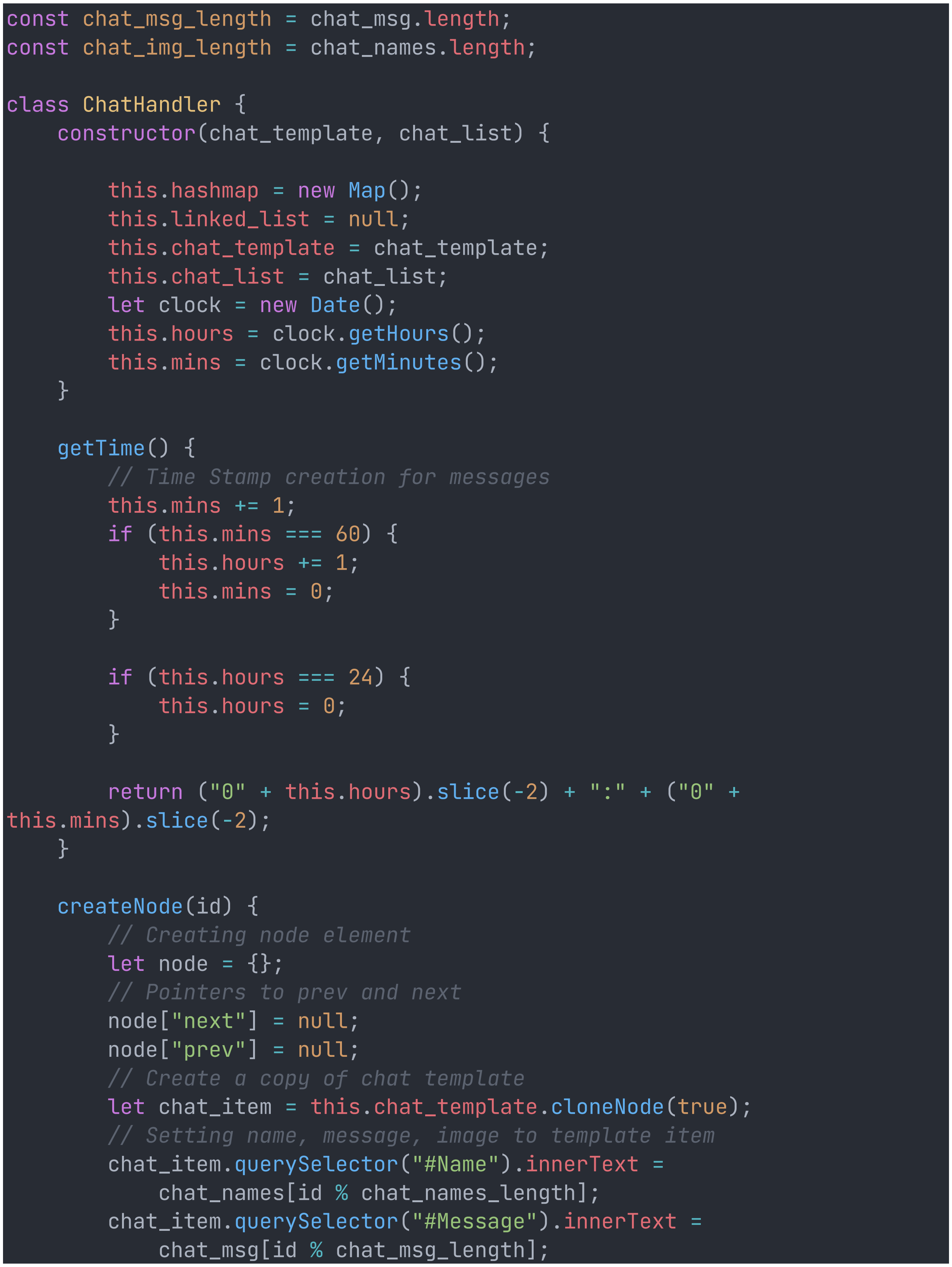
```
.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-body {
  -webkit-box-flex: 1;
  -webkit-flex-grow: 1;
  -ms-flex-positive: 1;
  flex-grow: 1;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-body>.ks-
message {
  font-size: 14px;
  color: #858585;
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  -js-display: flex;
  display: flex;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-body>.ks-
message>img {
  position: relative;
  top: -2px;
  width: 18px;
  height: 18px;
  margin-right: 5px;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-body>.ks-
name {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  -js-display: flex;
  display: flex;
  margin-bottom: 4px;
  -webkit-box-pack: justify;
  -webkit-justify-content: space-between;
  -ms-flex-pack: justify;
  justify-content: space-between;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item>a>.ks-body>.ks-
name>.ks-datetime {
  text-transform: uppercase;
  font-size: 10px;
  font-weight: 400;
  color: #858585;
  position: relative;
  top: 3px;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item.ks-active {
  background: rgba(0, 24, 240, 0.1);
  color: #333;
  position: relative;
}

.ks-messenger .ks-discussions>.ks-body .ks-items>.ks-item.ks-
active::before {
  content: "";
  width: 4px;
  height: 100%;
  background-color: #008cba;
  display: block;
  position: absolute;
  top: 0;
  left: 0;
}
```

<u>Scipt.js</u>

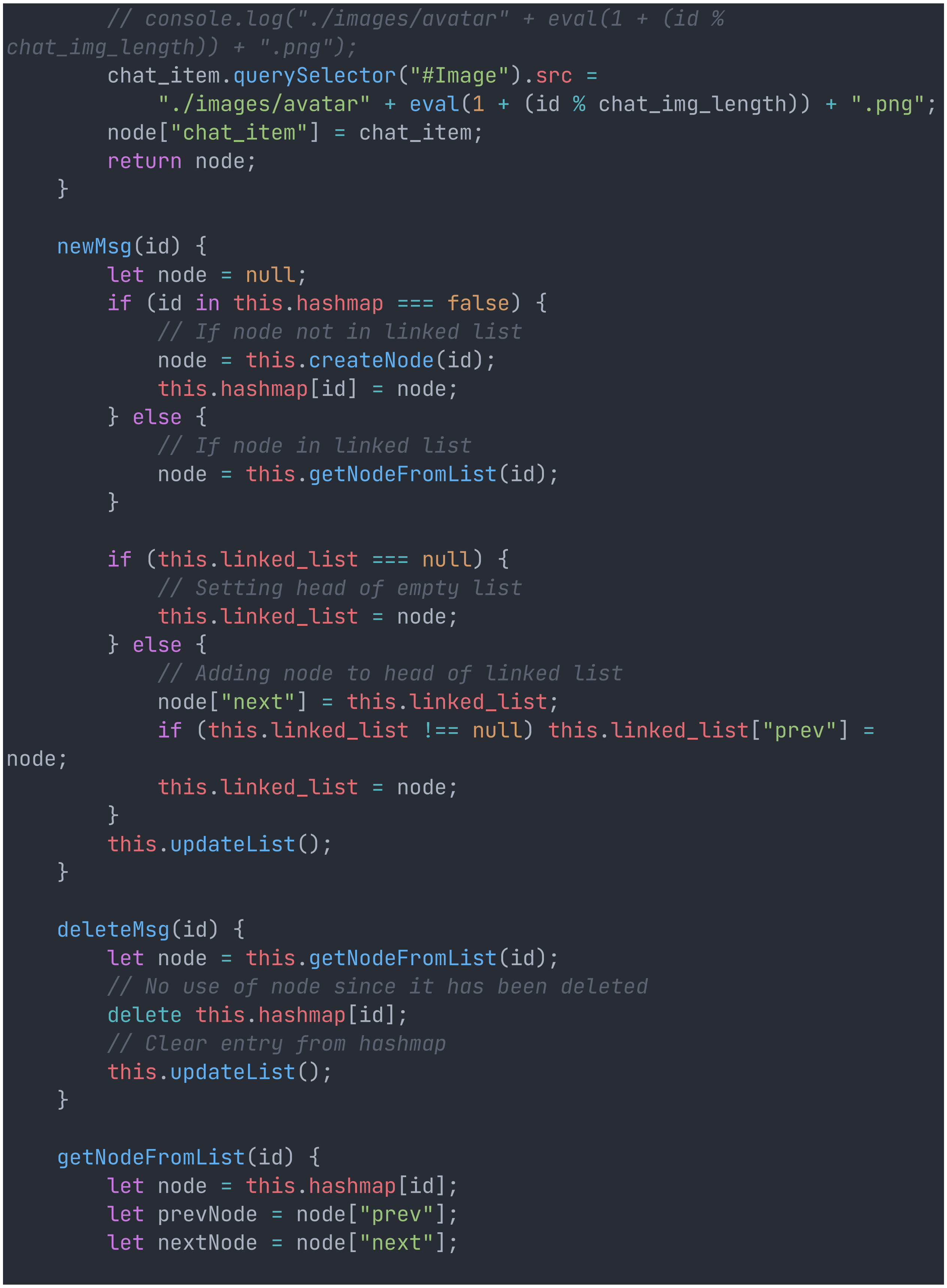
```
import { ChatHandler, chat_names } from "./ChatHandler.js";

onload = function () {
  const chatlist = document.getElementById("chat-list");
  const add = document.getElementById("generate-step");
  const text = document.getElementById("temptext");

  const templates = document.getElementsByTagName("template")[0];
  const chat_item = templates.content.querySelector("li");

  const chatHandler = new ChatHandler(chat_item, chatlist);
  let chats = [];

  add.onclick = function () {
    if (Math.random() > 0.75 && chats.length > 0) {
      let index = Math.floor(Math.random() * chats.length);
      let idToDelete = chats[index];
      chatHandler.deleteMsg(idToDelete);
      text.innerHTML =
        "Deleted message from " +
        chat_names[idToDelete] +
        "<br>" +
        text.innerHTML;
      chats.splice(index, 1);
    } else {
      let idOfMsg = Math.floor(Math.random() * 7);
      if (chats.includes(idOfMsg) === false) {
        chats.push(idOfMsg);
      }
      chatHandler.newMsg(idOfMsg);
      text.innerHTML =
        "New message from " + chat_names[idOfMsg] + "<br>" +
text.innerHTML;
    }
  };
};
```

Chathandler.js

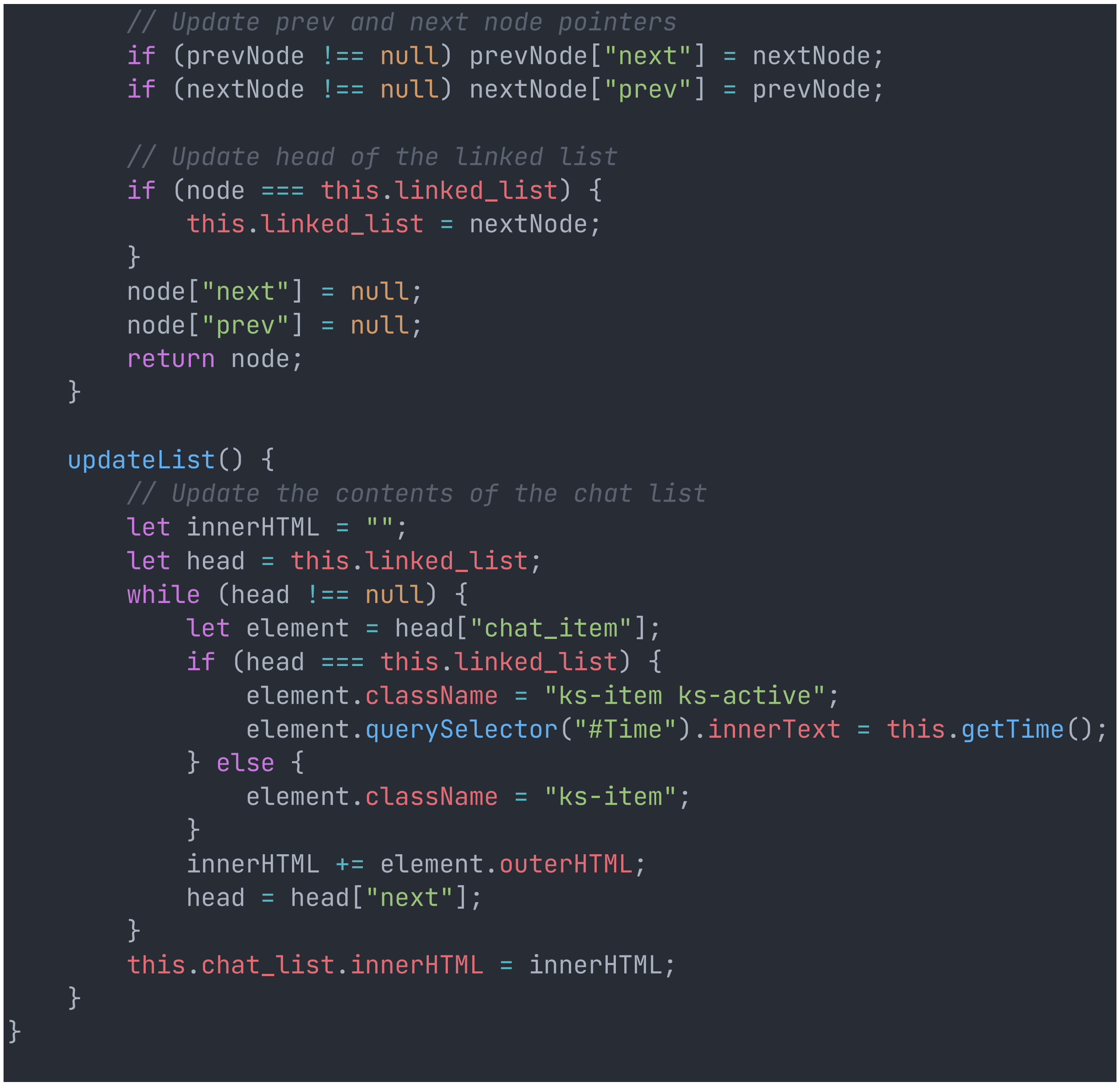
```
// ChatHandler.js

export { ChatHandler, chat_names };

const chat_names = [
    "Aman Kshetri",
    "Raj Sah",
    "Apurva K",
    "Saroj Bhagat",
    "Pinky Jain",
    "Tej Mahesh",
    "Vinay Kumar",
];

const chat_names_length = chat_names.length;
const chat_msg = [
    "Hey lets catch up today for lunch...",
    "Why didn't he come and talk to me...",
    "Perfect, I am really glad to hear that...",
    "This is what I understand you're telling...",
    "I'm sorry, I don't have the info on that...",
];
```



```
const chat_msg_length = chat_msg.length;
const chat_img_length = chat_names.length;

class ChatHandler {
    constructor(chat_template, chat_list) {

        this.hashmap = new Map();
        this.linked_list = null;
        this.chat_template = chat_template;
        this.chat_list = chat_list;
        let clock = new Date();
        this.hours = clock.getHours();
        this.mins = clock.getMinutes();
    }

    getTime() {
        // Time Stamp creation for messages
        this.mins += 1;
        if (this.mins === 60) {
            this.hours += 1;
            this.mins = 0;
        }

        if (this.hours === 24) {
            this.hours = 0;
        }

        return ("0" + this.hours).slice(-2) + ":" + ("0" +
this.mins).slice(-2);
    }

    createNode(id) {
        // Creating node element
        let node = {};
        // Pointers to prev and next
        node["next"] = null;
        node["prev"] = null;
        // Create a copy of chat template
        let chat_item = this.chat_template.cloneNode(true);
        // Setting name, message, image to template item
        chat_item.querySelector("#Name").innerText =
            chat_names[id % chat_names_length];
        chat_item.querySelector("#Message").innerText =
            chat_msg[id % chat_msg_length];
```



```
        // console.log("./images/avatar" + eval(1 + (id %
chat_img_length)) + ".png");
        chat_item.querySelector("#Image").src =
            "./images/avatar" + eval(1 + (id % chat_img_length)) + ".png";
        node["chat_item"] = chat_item;
        return node;
    }

    newMsg(id) {
        let node = null;
        if (id in this.hashmap === false) {
            // If node not in linked list
            node = this.createNode(id);
            this.hashmap[id] = node;
        } else {
            // If node in linked list
            node = this.getNodeFromList(id);
        }

        if (this.linked_list === null) {
            // Setting head of empty list
            this.linked_list = node;
        } else {
            // Adding node to head of linked list
            node["next"] = this.linked_list;
            if (this.linked_list !== null) this.linked_list["prev"] =
node;
            this.linked_list = node;
        }
        this.updateList();
    }

    deleteMsg(id) {
        let node = this.getNodeFromList(id);
        // No use of node since it has been deleted
        delete this.hashmap[id];
        // Clear entry from hashmap
        this.updateList();
    }

    getNodeFromList(id) {
        let node = this.hashmap[id];
        let prevNode = node["prev"];
        let nextNode = node["next"];
```

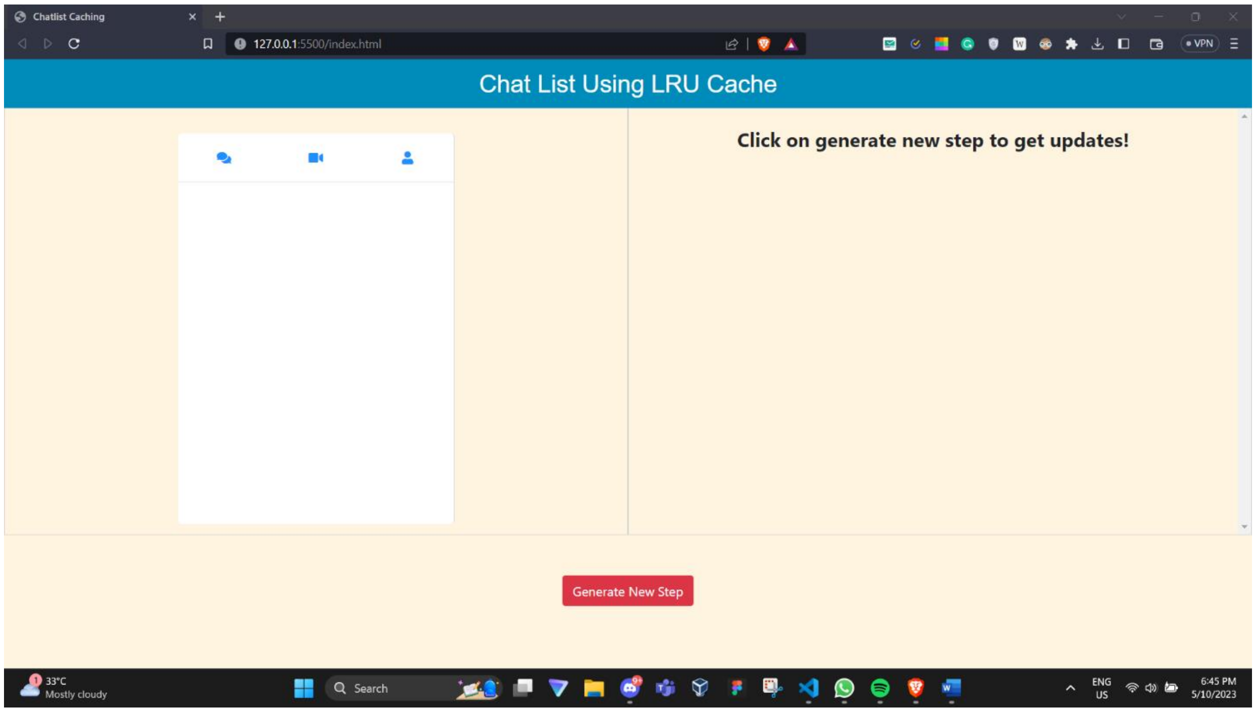


```
        // Update prev and next node pointers
        if (prevNode !== null) prevNode["next"] = nextNode;
        if (nextNode !== null) nextNode["prev"] = prevNode;

        // Update head of the linked list
        if (node === this.linked_list) {
            this.linked_list = nextNode;
        }
        node["next"] = null;
        node["prev"] = null;
        return node;
    }

    updateList() {
        // Update the contents of the chat list
        let innerHTML = "";
        let head = this.linked_list;
        while (head !== null) {
            let element = head["chat_item"];
            if (head === this.linked_list) {
                element.className = "ks-item ks-active";
                element.querySelector("#Time").innerText = this.getTime();
            } else {
                element.className = "ks-item";
            }
            innerHTML += element.outerHTML;
            head = head["next"];
        }
        this.chat_list.innerHTML = innerHTML;
    }
}
```

5) Output Screenshots

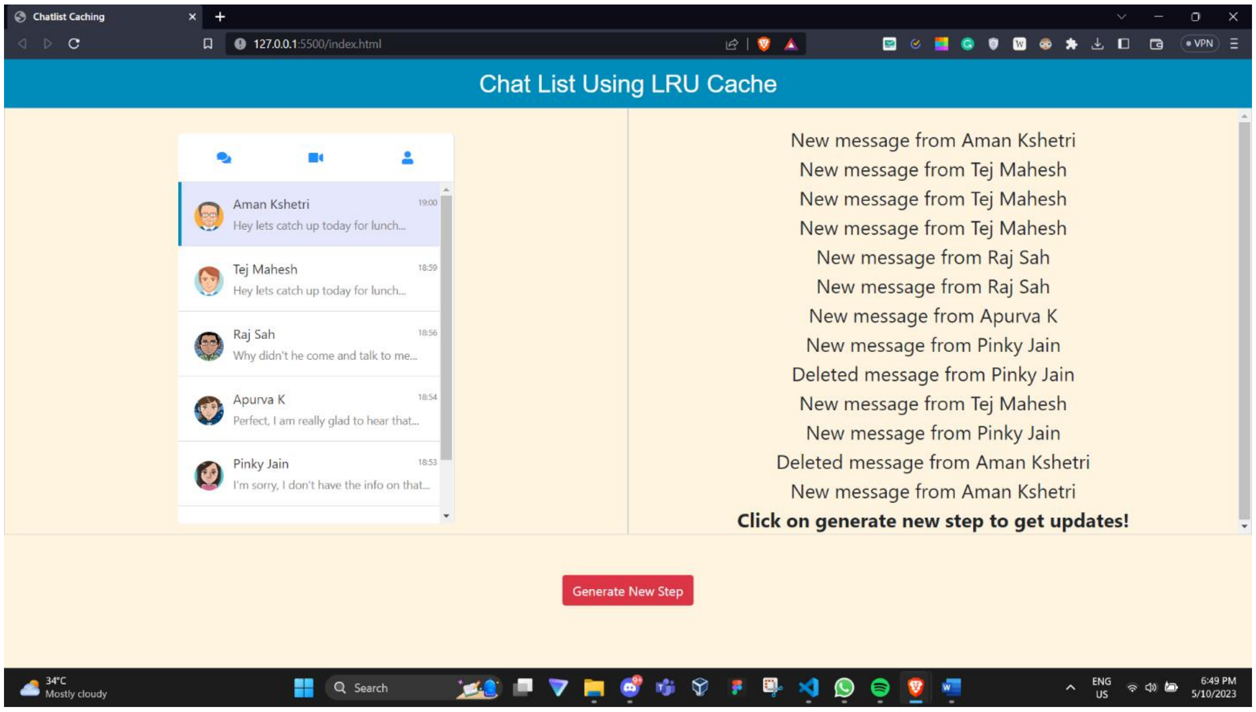
The final output of the above code is given below:

Before Generating New Steps:



Click on generate new step to get updates!

After Generating New Steps:



Aman Kshetri	Hey lets catch up today for lunch...	19:00
Tej Mahesh	Hey lets catch up today for lunch...	18:59
Raj Sah	Why didn't he come and talk to me...	18:56
Apurva K	Perfect, I am really glad to hear that...	18:54
Pinky Jain	I'm sorry, I don't have the info on that...	18:53

```
New message from Aman Kshetri
New message from Tej Mahesh
New message from Tej Mahesh
New message from Tej Mahesh
New message from Raj Sah
New message from Raj Sah
New message from Apurva K
New message from Pinky Jain
Deleted message from Pinky Jain
New message from Tej Mahesh
New message from Pinky Jain
Deleted message from Aman Kshetri
New message from Aman Kshetri
```

Click on generate new step to get updates!