
Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
DSA miniProject
A.Y. 2025-26

**Title: Browser Management using stacks in DSA**

**Sustainability Goal : To develop a memory-efficient and scalable navigation logic using a doubly linked list, designed with modularity for long-term maintenance and reuse.**

**Domain: Data Structures and Algorithms**

**Member: Pradnyesh Patil**
**D10B 45**

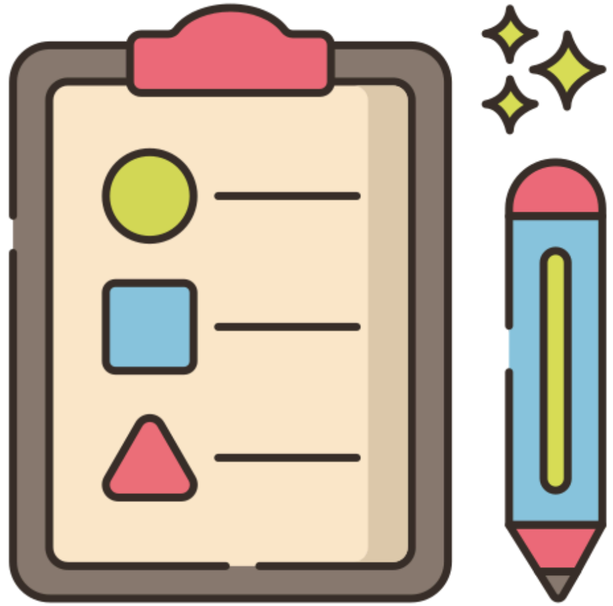
**Mentor Name : Kajal Jewani**


1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
THE GLOBAL GOALS
For Sustainable Development
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

# Content

# Introduction to Project

- **Web browsers are essential digital tools, and their ability to navigate back and forward is a key feature powered by efficient data structures. This project demonstrates how a doubly linked list can be used to create a fast and reliable browser history management system.**



- **Our goal is to apply core concepts from Data Structures and Algorithms (DSA) to build a simplified yet powerful model of this real-world software feature, showcasing how theoretical knowledge solves practical problems.**

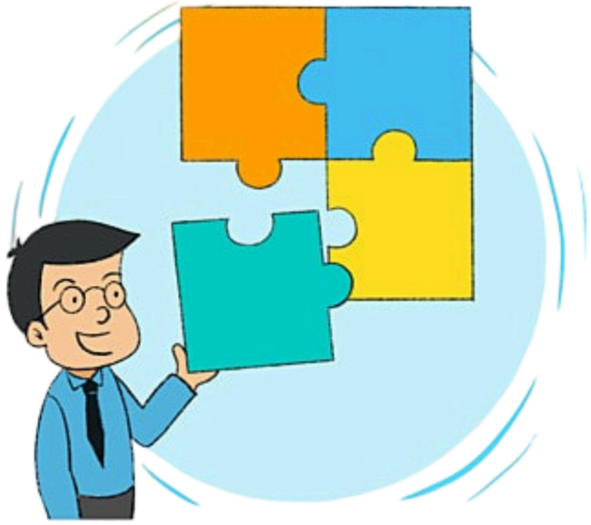
# Problem Statement

- **Modern web browsers require an efficient mechanism to manage user navigation history. While simple data structures like arrays can be slow for this task, a doubly linked list provides a highly effective solution. This structure allows for constant-time (O(1)) back and forward operations because each page is directly linked to its predecessor and successor.**

- **This project aims to design and implement a simplified browser history manager using a doubly linked list, demonstrating an efficient, real-world application of this core data structure.**

# Objectives of the project

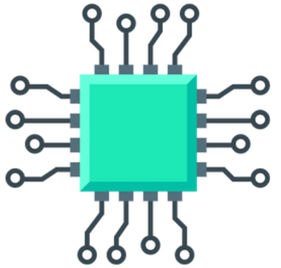
- To implement core navigation features, allowing users to visit new pages and move backward and forward through their history with O(1) time complexity.
- To build the history mechanism using a doubly linked list as the core data structure for efficient traversal and modification.
- To provide a comprehensive set of history management functions, including page deletion and clearing the entire history.
- To extend the system's functionality by integrating a third-party API for real-time URL shortening.



# Requirements of the system (Hardware, software)

**Hardware:**

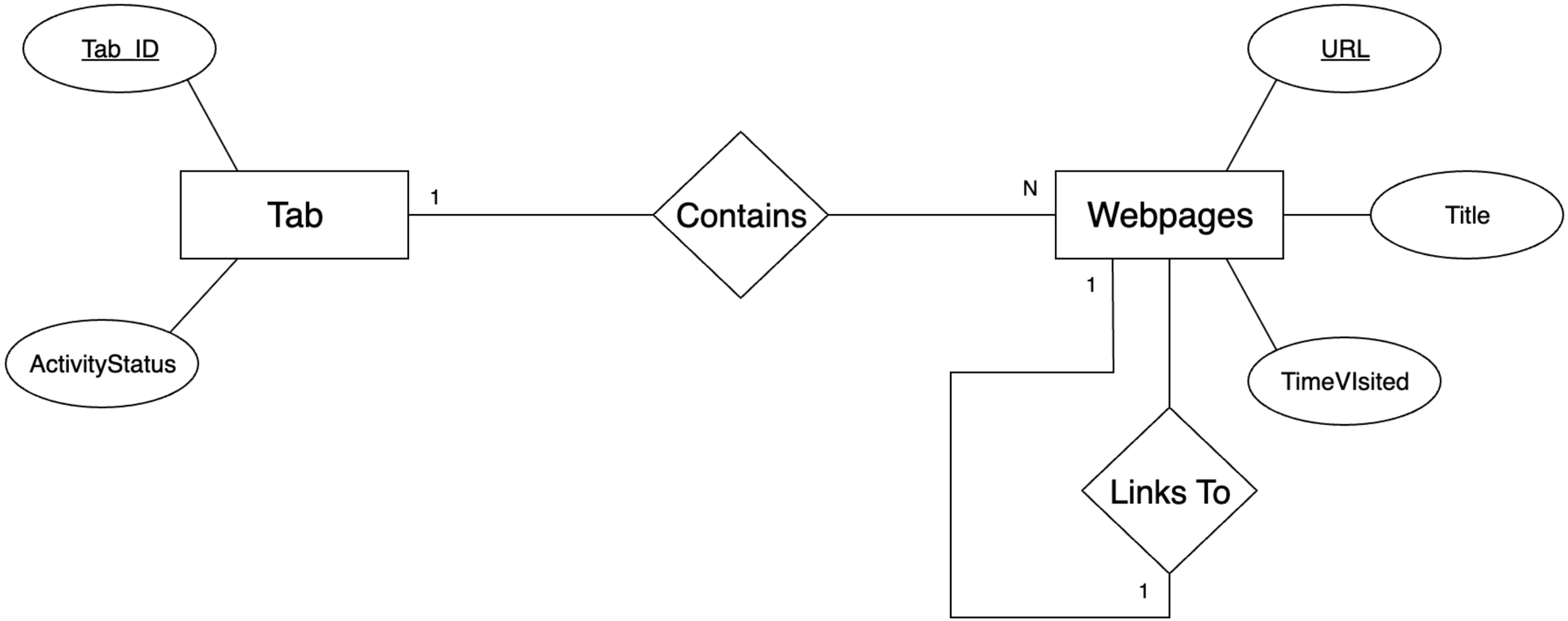
- **Basic computer with 4 GB RAM**
- **50 MB free storage**
- **and keyboard/mouse.**



**Software:**

- **Operating System: Windows/Linux/macOS**
- **Compiler: GCC / Turbo C**
- **Libraries: The libcurl and ncurses development libraries must be installed.**
- **IDE (Optional): Visual Studio Code, Code::Blocks, or any standard text editor.**
- **Other: An active internet connection (for the URL shortening feature).**

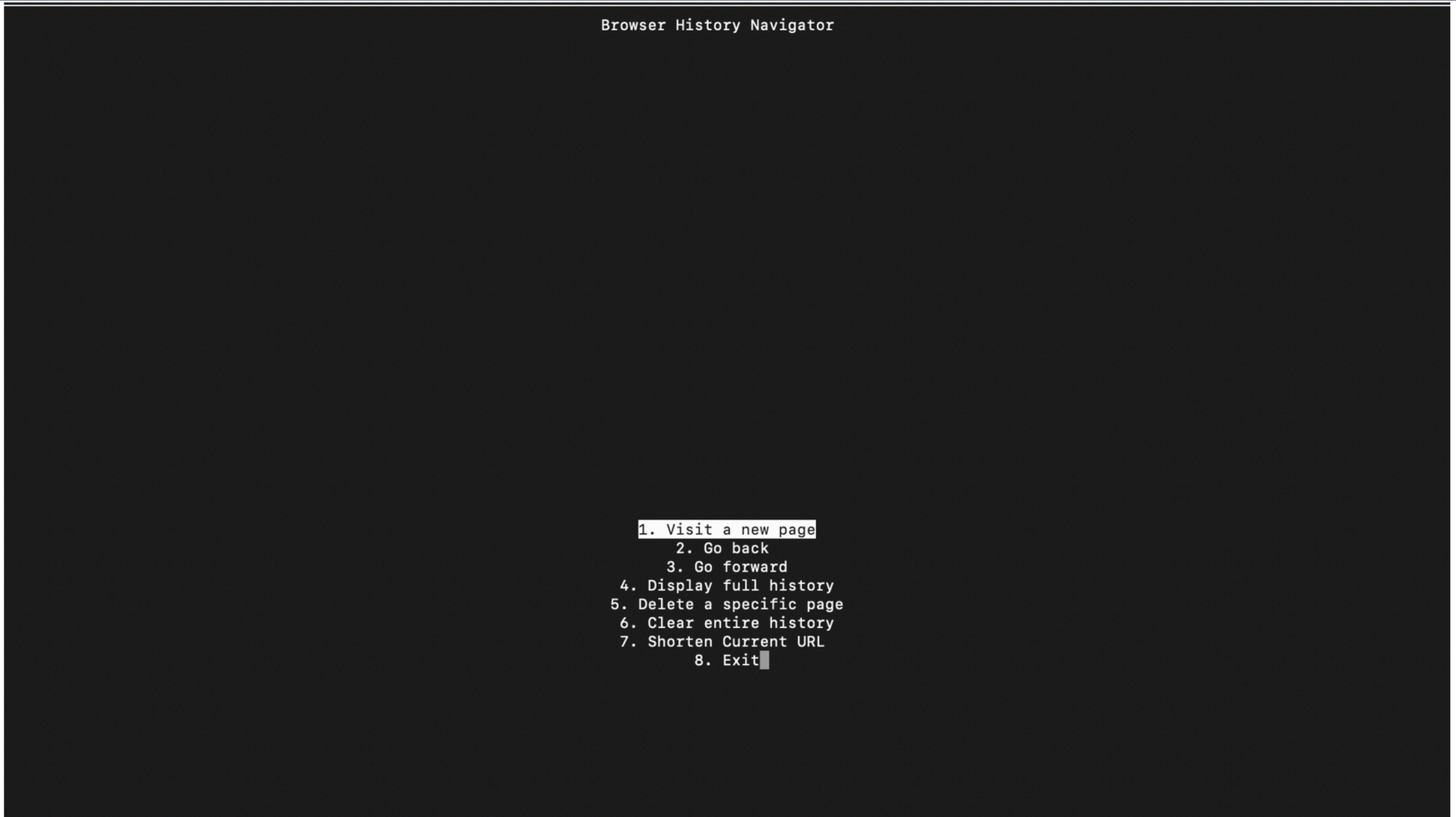
# ER diagram of the proposed system



# Front End

- **Interface: An interactive Text-based User Interface (TUI) built with the ncurses library, providing a graphical, full-screen experience within the terminal.**
- **Input: User commands are handled through real-time keyboard input, including arrow keys for menu navigation and the Enter key for selection.**
- **Output: A dynamic, bordered menu with highlighting to indicate the user's current selection, providing a clear and user-friendly interface.**

# Front End


Browser History Navigator
1. Visit a new page
2. Go back
3. Go forward
4. Display full history
5. Delete a specific page
6. Clear entire history
7. Shorten Current URL
8. Exit

# Implementation

- **Data Structure: A doubly linked list is used as the core data structure to manage the browsing history, allowing for efficient O(1) time complexity for navigation operations.**
- **Core Functions: The system is built around key functions like visitNewPage(), goBack(), goForward(), deleteParticularPage(), and displayHistory().**
- **Frontend Interface: An interactive Text-based User Interface (TUI), built with the ncurses library, handles user interaction via arrow key navigation and a highlighted menu.**
- **API Integration: The libcurl library is integrated to communicate with the TinyURL API, providing real-time URL shortening functionality.**


```c
You, 23 hours ago | 1 author (You)
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>
#include <ncurses.h>

#define MAX_URL_LENGTH 3000

You, 3 days ago | 1 author (You)
struct Node {
    char url[MAX_URL_LENGTH];
    struct Node * next;
    struct Node * prev;
};

You, 3 days ago | 1 author (You)
struct browserHistory {
    struct Node * head;
    struct Node * tail;
    struct Node * current;
};


//function 1: visiting new page loml
void visitNewPage(struct browserHistory* history, const char* url) {
    struct Node * newNode;
    newNode = (struct Node *)malloc(sizeof(struct Node));

    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }

    strncpy(newNode->url, url, MAX_URL_LENGTH - 1);
    newNode->next = NULL;
    newNode->prev = NULL;
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void visitNewPage(struct browserHistory* history, const char* u
    newNode->prev = NULL;
    newNode->url[MAX_URL_LENGTH - 1] = '\0';        //  \0 work

    //history is completely empty -> no tabs are opened
    if(history -> current == NULL) {
        history -> head = newNode;
        history -> tail = newNode;
        history -> current = newNode;
        printf("Visited a new page: %s\n", url);
        return;
    }

    //A -> B -> C -> D       A -> B -> X            C AND D ARE
    if(history -> current -> next != NULL) {
        printf("Deleting forward history\n");
        struct Node * temp = history -> current -> next;

        while(temp != NULL) {
            struct Node * nodeToDelete = temp;
            temp = temp -> next;
            printf("%s Page deleted\n", nodeToDelete -> url);
            free(nodeToDelete);
        }
    }
    history -> current -> next = newNode;
    newNode -> prev = history -> current;

    history -> current = newNode;
    history -> tail = newNode;

    printf("Visited new page: %s\n", url);
}

//function 2: going back
void goBack(struct browserHistory* history) {
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void goBack(struct browserHistory* history) {
    if (history->current == NULL || history->current->prev == NULL) {
        printf("Cannot go back. This is the oldest page in the history.\n");
        return;
    } else {
        history -> current = history -> current -> prev;
        printf("Going Back, Current Page now is: %s\n", history -> current -> url);
    }
}

//function 3: going forward
void goForward(struct browserHistory* history) {
    if(history -> current == NULL || history -> current -> next == NULL) {
        printf("Cant go Forward, This is the Latest Page\n");
        return;
    } else {
        history -> current = history -> current -> next;
        printf("Going Forward, Current Page now is: %s\n", history -> current -> url);
    }
}


//function 4: displays history
void displayHistory(struct browserHistory* history) {
    printf("-------BROWSER HISTORY-------\n");
    if(history -> head == NULL){
        printf("History is Empty\n");
        printf("-----------------------------\n");
        return;
    }
    struct Node * ptr = history -> head;
    while (ptr != NULL) {
        printf("%s", ptr->url);

        if(ptr == history -> current) {
            printf("    <--- YOU ARE HERE");
```

doublyLLhist.c | .gitignore 0 | doublyLLhist02.c 0 | tui_test.c 0 | api_test.c

.linkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void displayHistory(struct browserHistory* history) {
        while (ptr != NULL) {
            if(ptr == history -> current) {
                printf("    <--- YOU ARE HERE");
            }
            printf("\n");
            ptr = ptr->next;
        }
    printf("-----------------------------\n");
}


//function 5: deletes entire history
//didnt understand this achese ;-;
void deleteEntireHistory(struct browserHistory* history) {
    struct Node * ptr = history -> head;
    struct Node * nodeToDelete;

    while(ptr != NULL) {
        nodeToDelete = ptr;
        ptr = ptr -> next;
        free(nodeToDelete);
    }
    history -> head = NULL;
    history -> tail = NULL;
    history -> current = NULL;

    printf("\nBrowser history cleared successfully.\n");
}

void deleteParticularPage(struct browserHistory* history, const char* urlToDelete) {

    struct Node * ptr = history -> head;
    struct Node * nodeToDelete;

    //strcmp returns 0 when the 2 strings are same, it is not like true false
```

```c
void deleteParticularPage(struct browserHistory* history, const char* urlToDelete) {
    //strcmp returns 0 when the 2 strings are same, it is not like true false
    while(ptr != NULL && strcmp(ptr -> url, urlToDelete) != 0) {
        ptr = ptr -> next;
    }

    //1.
    if (ptr == NULL) {
        printf("Page not found: %s\n", urlToDelete);
        return;
    }

    //2.
    //move the bookmark to the previous page.
    //It prevents the current pointer from pointing to something that no longer exists.
    if (history -> current == ptr) {
        history -> current = ptr -> prev;
    }

    if(ptr == history -> head) {
        history -> head = ptr -> next;
        if(history -> head != NULL) {
            history -> head -> prev = NULL;
        }
    }

    else if (ptr == history -> tail) {
        history -> tail = ptr -> prev;
        history -> tail -> next = NULL;
    }

    else {
        ptr->prev->next = ptr->next;
        ptr->next->prev = ptr->prev;
    }
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void deleteParticularPage(struct browserHistory* history, const char* urlToDelete) {

    if (history->head == NULL) {
        history->tail = NULL;
        history->current = NULL;
    }

    printf("Successfully deleted page: %s\n", ptr->url);
    free(ptr);
}


//integrating API in the project xD
// The callback function
size_t write_callback(void *ptr, size_t size, size_t nmemb, void *userdata) {
    printf("--> Shortened URL: %s\n", (char *)ptr);
    return size * nmemb;
}

// The new function to call the API
void shortenUrl(const char* longUrl) {
    if (longUrl == NULL) return;
    CURL *curl;
    char apiUrl[MAX_URL_LENGTH + 50];
    sprintf(apiUrl, "https://tinyurl.com/api-create.php?url=%s", longUrl);

    curl = curl_easy_init();
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL, apiUrl);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_callback);
        curl_easy_perform(curl);
        curl_easy_cleanup(curl);
    }
}

void shortenUrlTUI(struct browserHistory* history) {
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void shortenUrlTUI(struct browserHistory* history) {
    if (history->current == NULL) {
        mvprintw(LINES - 2, 1, "No current page to shorten. Press any key...");
        getch();
        return;
    }


    endwin();
    printf("Contacting API...\n");
    shortenUrl(history->current->url);       You, 23 hours ago • removed some l

    printf("\nPress any key to return to the menu...");
    getch();
}


void draw_menu(int highlight) {
    clear(); // Clear the screen
    int height, width;
    getmaxyx(stdscr, height, width); // Get screen dimensions

    mvprintw(1, (width - 25) / 2, "Browser History Navigator"); // Title
    box(stdscr, 0, 0); // Draw a border

    char* choices[] = {
        "1. Visit a new page",
        "2. Go back",
        "3. Go forward",
        "4. Display full history",
        "5. Delete a specific page",
        "6. Clear entire history",
        "7. Shorten Current URL",
        "8. Exit"
    };
    int num_choices = 8;
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
void draw_menu(int highlight) {
    char* choices[] = {
    int num_choices = 8;

    for (int i = 0; i < num_choices; i++) {
        int y = (height / 2) - (num_choices / 2) + i;
        int x = (width / 2) - (strlen(choices[i]) / 2);

        if (i == highlight) {
            attron(A_REVERSE);
        }
        mvprintw(y, x, "%s", choices[i]);
        if (i == highlight) {
            attroff(A_REVERSE);
        }
    }
    refresh();
}

int main() {
    struct browserHistory myBrowser = {NULL, NULL, NULL};
    char url[MAX_URL_LENGTH];


    initscr();
    cbreak();
    noecho();
    keypad(stdscr, TRUE);

    int highlight = 0;
    int choice = 0;


    while(1) {
        draw_menu(highlight);
        int input = getch();
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
int main() {
    while(1) {
        int input = getch();
        switch(input) {
            case KEY_UP:
                highlight--;
                if (highlight < 0) highlight = 7;
                break;
            case KEY_DOWN:
                highlight++;
                if (highlight > 7) highlight = 0;
                break;
            case 10:
                choice = highlight + 1;
                break;
            default:
                break;
        }


        if (choice != 0) {

            if (choice == 1 || choice == 4 || choice == 5 || choice == 7) {
                endwin();
                printf("\n");

                switch(choice) {
                    case 1:
                        printf("Enter the URL to visit: ");
                        scanf("%s", url);
                        visitNewPage(&myBrowser, url);
                        break;
                    case 4:
                        displayHistory(&myBrowser);
                        break;
                    case 5:
```

LinkedLists > C doublyLLhist.c > shortenUrlTUI(browserHistory *)

```c
int main() {
    while(1) {
        if (choice != 0) {
            if (choice == 1 || choice == 4 || choice == 5 || choice == 7) {

                initscr();
                cbreak();
                noecho();
                keypad(stdscr, TRUE);

            } else {

                switch(choice) {
                    case 2: goBack(&myBrowser); break;
                    case 3: goForward(&myBrowser); break;
                    case 6: deleteEntireHistory(&myBrowser); break;
                }

                mvprintw(LINES - 2, 1, "Action complete. Press any key...");
                getch();
            }

            if (choice == 8) { // Exit condition
                break;
            }
            choice = 0; // Reset choice
        }
    }


    endwin();
    deleteEntireHistory(&myBrowser);
    printf("Exited successfully.\n");

    return 0;
}
```

# Output:

```
pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air ~ % cd desktop
pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air desktop % cd dsaxTry
pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air dsaxTry % cd LinkedLists
pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air LinkedLists % export LDFLAGS="-L/usr/local/opt/ncurses/lib -L/usr/local/opt/curl/lib"
export CPPFLAGS="-I/usr/local/opt/ncurses/include -I/usr/local/opt/curl/include"

pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air LinkedLists % gcc doublyLLhist.c -o browser -lcurl -lncurses
pradnyeshsheetalprashantpatil@Pradnyeshs-MacBook-Air LinkedLists %
./browser



Enter the URL to visit: https://github.com/pradsgoat7/DSAminiPro/blob/main/doublyLLhist.c
Visited a new page: https://github.com/pradsgoat7/DSAminiPro/blob/main/doublyLLhist.c
```

```
Press any key to return to the menu...
Enter the URL to visit:
https://youtu.be/ilNt2bikxDI?list=RDilNt2bikxDI
Visited new page: https://youtu.be/ilNt2bikxDI?list=RDilNt2bikxDI


Press any key to return to the menu...
Enter the URL to visit: https://chatgpt.com/
Visited new page: https://chatgpt.com/
```

# Output:

```
Press any key to return to the menu...
-------BROWSER HISTORY-------
https://github.com/pradsgoat7/DSAminiPro/blob/main/doublyLLhist.c
https://youtu.be/ilNt2bikxDI?list=RDilNt2bikxDI
https://chatgpt.com/    <--- YOU ARE HERE
----------------------------


Press any key to return to the menu...
Enter the URL to delete: https://chatgpt.com/
Successfully deleted page: https://chatgpt.com/
```

```
Contacting API...return to the menu...
--> Shortened URL: https://tinyurl.com/2bcadmd6



-------BROWSER HISTORY-------
https://github.com/pradsgoat7/DSAminiPro/blob/main/doublyLLhist.c    <--- YOU ARE HERE
https://youtu.be/ilNt2bikxDI?list=RDilNt2bikxDI
----------------------------

Press any key to return to the menu...
-------BROWSER HISTORY-------
https://github.com/pradsgoat7/DSAminiPro/blob/main/doublyLLhist.c    <--- YOU ARE HERE
https://youtu.be/ilNt2bikxDI?list=RDilNt2bikxDI
----------------------------
```

# Gantt Chart

**Phase 1: Core Logic Development (Days 1 - 3)**

- Designed the doubly linked list data structure.
- Implemented all core history functions (visit, back, forward, delete, display).

**Phase 2: Advanced Feature Integration (Days 4 - 5)**

- Integrated the libcurl library to connect with the TinyURL API.
- Implemented an interactive Text-based User Interface (TUI) using ncurses.

**Phase 3: Finalisation & Deployment (Days 6 - 7)**

- Completed final testing, debugging, and code cleanup.
- Created the project presentation and uploaded the final code to GitHub.

Gantt Chart

# Conclusion

**Conclusion:**

- **This project successfully demonstrates a real-world application of data structures by building a robust browser history manager in C.**

- **The use of a doubly linked list provides efficient O(1) navigation, while the interactive TUI frontend and live API integration showcase how core logic can be enhanced with modern libraries to create a complete and feature-rich application.**

# References


- GeeksforGeeks - Doubly Linked List A comprehensive web resource for the theory and C implementation of the core data structure used in this project.
  https://www.geeksforgeeks.org/doubly-linked-list/

- NCURSES Programming HOWTO A detailed guide on programming Text-based User Interfaces (TUIs) in C using the ncurses library.
  https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/

- libcurl Official Website - C Examples The official documentation and example code for libcurl, the library used to perform API requests and handle internet communication.
  https://curl.se/libcurl/c/example.html