****

**Department of Creative Technology**

**Course Code**

CS-214

**Subject**

Data Structures & Algorithms

**Instructor:**

Mr. Muhammad Adnan Aslam

**Group Members:** Muhammad Faseeh (221839)

Danial Saleem (221825)

Section: BSSE-III- A

Date of Submission: 27/12/2023

**Project Name: Duplicate Words Finder**

**Introduction:**

The purpose of this project is to identify and manage duplicate content (Words) within chosen text files. We use the SFML library to create the GUI of our program. The program identifies duplicate words between two text files and displays them on the screen with a Beautiful User Interface. It also allows the user to keep words remaining the same or delete them.

**Problem Statement:**

A common challenge that we encountered is managing duplicate content especially when we have a large amount of data. At this moment this is time-consuming to find duplicate content and then delete them one by one.

**Solution:**

1. **Linked list Representation:**

We implement a linked list data structure to manage file content. Linked list stores file content word by word and then identifies duplicates, and displays them on screen using the SFML library. Then the program deletes them by consent of the user by eliminating those words from one of the files.

**Concepts Used/Topics:**

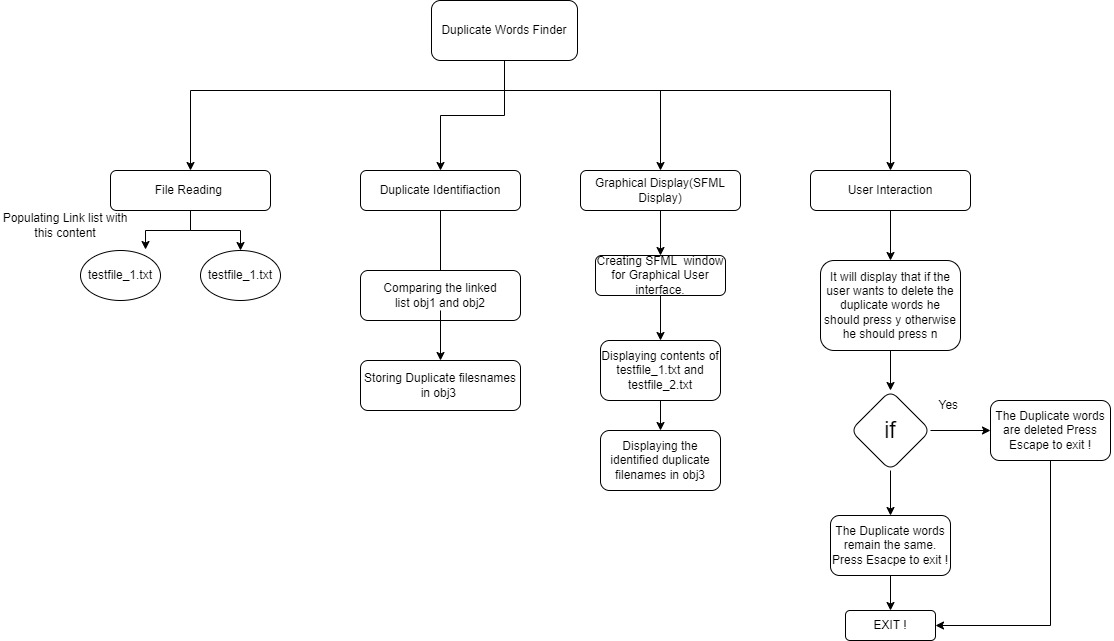
**Linked list**: Linked lists are used to store and manage data retrieved from files.

**File Handling**: we used file handling technique to read the content and write (edit) the content to delete duplicate contents.

**Assumption:**

When any one run this code, it is important to include SFML library to IDE project. It is important to include two files named as **testfile\_1** and **testfile\_2** in the a folder test file which is placed in same folder where source.cpp is placed.

**Data/Flow Diagram:**



**Code:**

#include <iostream>

#include <fstream>

#include "SFML/Graphics.hpp"

using namespace std;

class Node {

public:

string word;

Node\* next = nullptr;

};

class List {

public:

Node\* head;

List() {

head = nullptr;

}

void insert\_name(const string& p\_word) {

Node\* temp = new Node;

temp->word = p\_word;

temp->next = nullptr;

if (head == nullptr) {

head = temp;

}

else {

Node\* itemp = head;

while (itemp->next != nullptr) {

itemp = itemp->next;

}

itemp->next = temp;

}

}

void display\_obj(sf::RenderWindow& window, const std::string& title, float xPos, float yPos) {

Node\* temp = head;

sf::Font font;

if (!font.loadFromFile("arial.ttf")) {

cerr << "Error: Font not found." << endl;

return;

}

sf::Text titleText;

titleText.setFont(font);

titleText.setCharacterSize(30);

titleText.setFillColor(sf::Color::Red);

titleText.setPosition(xPos, yPos);

titleText.setString(title);

sf::Text fileListText;

fileListText.setFont(font);

fileListText.setCharacterSize(25);

fileListText.setFillColor(sf::Color::White);

float yOffset = 50.0f;

while (temp != nullptr) {

fileListText.setPosition(xPos + 100.f, yPos + yOffset);

fileListText.setString(temp->word);

window.draw(fileListText);

yOffset += 40.0f;

temp = temp->next;

}

window.draw(titleText);

}

};

void remove\_duplicate\_words(List& obj3) {

Node\* temp = obj3.head;

while (temp != nullptr) {

string wordToRemove = temp->word;

string filePath = "testfile/testfile\_2.txt";

ifstream inFile(filePath);

ofstream tempFile("testfile/temp.txt");

if (!inFile || !tempFile) {

cerr << "Error: File handling issue!" << endl;

return;

}

string word;

while (inFile >> word) {

if (word != wordToRemove) {

tempFile << word << " ";

}

}

inFile.close();

tempFile.close();

// Remove the original file

remove(filePath.c\_str());

// Rename temp file to original filename

rename("testfile/temp.txt", filePath.c\_str());

temp = temp->next;

}

}

void find\_duplicate(List obj1, List obj2, List& obj3) {

Node\* temp1 = obj1.head;

while (temp1 != NULL)

{

Node\* temp2 = obj2.head;

while (temp2 != NULL)

{

if (temp1->word == temp2->word)

{

obj3.insert\_name(temp1->word);

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void read\_file(const string& filePath, List& obj) {

ifstream file(filePath);

if (!file) {

cerr << "Error: Unable to open file " << filePath << endl;

return;

}

string filename;

while (file >> filename) {

obj.insert\_name(filename);

}

file.close();

}

int main() {

sf::RenderWindow window(sf::VideoMode(1300, 800), "File Contents Display");

List obj1, obj2, obj3;

read\_file("testfile/testfile\_1.txt", obj1);

read\_file("testfile/testfile\_2.txt", obj2);

find\_duplicate(obj1, obj2, obj3);

sf::Font font;

if (!font.loadFromFile("arial.ttf")) {

cerr << "Error: Font not found." << endl;

return EXIT\_FAILURE;

}

sf::Text promptText;

promptText.setFont(font);

promptText.setCharacterSize(30);

promptText.setFillColor(sf::Color::White);

promptText.setPosition(50.f, 600.f);

promptText.setString("Do you want to remove the duplicate files? (y/n): ");

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event)) {

if (event.type == sf::Event::Closed) {

window.close();

}

if (event.type == sf::Event::KeyPressed)

{

if (event.key.code == sf::Keyboard::Y)

{

remove\_duplicate\_words(obj3);

promptText.setString("Duplicate file has been deleted. Press Escape to Exit ! ");

}

if (event.key.code == sf::Keyboard::N)

{

promptText.setString("Duplicate file has remained same. Press Escape to Exit ! ");

}

if (event.key.code == sf::Keyboard::Escape) {

return 0;

}

}

}

window.clear();

obj1.display\_obj(window, "Contents of file 1:", 50.f, 50.f);

obj2.display\_obj(window, "Contents of file 2:", 50.f, 230.f);

obj3.display\_obj(window, "following are the duplicate words : ", 50.f, 450.f);

window.draw(promptText);

window.display();

}

return 0;

}

**Output:**

