#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define FILENAME "diary\_entries.txt"

// Function to get the current date and time

void get\_current\_datetime(char \*buffer, size\_t size) {

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

strftime(buffer, size, "%Y-%m-%d %H:%M:%S", &tm);

}

// Function to add an entry

void add\_entry() {

FILE \*file = fopen(FILENAME, "a");

if (!file) {

perror("Error opening file");

return;

}

char content[1024];

char datetime[20];

get\_current\_datetime(datetime, sizeof(datetime));

printf("Write your diary entry: ");

getchar(); // Consume leftover newline

fgets(content, sizeof(content), stdin);

content[strcspn(content, "\n")] = 0; // Remove newline

fprintf(file, "%s|%s\n", datetime, content);

fclose(file);

printf("Entry added successfully!\n");

}

// Function to view all entries

void view\_entries() {

FILE \*file = fopen(FILENAME, "r");

if (!file) {

perror("Error opening file");

return;

}

char line[1060];

printf("\n--- Diary Entries ---\n");

while (fgets(line, sizeof(line), file)) {

char \*datetime = strtok(line, "|");

char \*content = strtok(NULL, "|");

if (datetime && content) {

printf("Date: %s\nContent: %s\n\n", datetime, content);

}

}

fclose(file);

}

// Function to search entries by keyword

void search\_entries() {

FILE \*file = fopen(FILENAME, "r");

if (!file) {

perror("Error opening file");

return;

}

char keyword[100];

printf("Enter keyword to search: ");

getchar(); // Consume leftover newline

fgets(keyword, sizeof(keyword), stdin);

keyword[strcspn(keyword, "\n")] = 0; // Remove newline

char line[1060];

int found = 0;

printf("\n--- Search Results ---\n");

while (fgets(line, sizeof(line), file)) {

if (strstr(line, keyword)) {

found = 1;

char \*datetime = strtok(line, "|");

char \*content = strtok(NULL, "|");

if (datetime && content) {

printf("Date: %s\nContent: %s\n\n", datetime, content);

}

}

}

if (!found) {

printf("No entries found containing '%s'.\n", keyword);

}

fclose(file);

}

// Function to delete all entries

void delete\_entries() {

printf("Are you sure you want to delete all entries? (y/n): ");

char choice;

scanf(" %c", &choice);

if (choice == 'y' || choice == 'Y') {

FILE \*file = fopen(FILENAME, "w");

if (file) {

fclose(file);

printf("All entries deleted successfully.\n");

} else {

perror("Error clearing file");

}

} else {

printf("Deletion canceled.\n");

}

}

// Function to update a specific entry

void update\_entry() {

FILE \*file = fopen(FILENAME, "r");

if (!file) {

perror("Error opening file");

return;

}

FILE \*temp = fopen("temp.txt", "w");

if (!temp) {

perror("Error opening temporary file");

fclose(file);

return;

}

char datetime[20], content[1024], line[1060];

char datetime\_to\_update[20];

int found = 0;

printf("Enter the date and time of the entry to update (format: YYYY-MM-DD HH:MM:SS): ");

getchar(); // Consume leftover newline

fgets(datetime\_to\_update, sizeof(datetime\_to\_update), stdin);

datetime\_to\_update[strcspn(datetime\_to\_update, "\n")] = 0; // Remove newline

while (fgets(line, sizeof(line), file)) {

sscanf(line, "%19[^|]|%1023[^\n]", datetime, content);

if (strcmp(datetime, datetime\_to\_update) == 0) {

found = 1;

printf("Current content: %s\n", content);

printf("Enter new content: ");

fgets(content, sizeof(content), stdin);

content[strcspn(content, "\n")] = 0; // Remove newline

fprintf(temp, "%s|%s\n", datetime, content);

} else {

fprintf(temp, "%s", line);

}

}

fclose(file);

fclose(temp);

if (found) {

remove(FILENAME);

rename("temp.txt", FILENAME);

printf("Entry updated successfully.\n");

} else {

printf("No entry found with the given date and time.\n");

remove("temp.txt");

}

}

// Main function

int main() {

int choice;

do {

printf("\n--- Personal Diary ---\n");

printf("1. Add Entry\n");

printf("2. View Entries\n");

printf("3. Search Entries\n");

printf("4. Delete All Entries\n");

printf("5. Update Entry\n");

printf("0. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

add\_entry();

break;

case 2:

view\_entries();

break;

case 3:

search\_entries();

break;

case 4:

delete\_entries();

break;

case 5:

update\_entry();

break;

case 0:

printf("Exiting...\n");

break;

default:

printf("Invalid choice! Please try again.\n");

}

} while (choice != 0);

return 0;

}

OUTPUT:



