CODE:

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

class Task {

public:

string description;

string dueDate;

bool isCompleted;

Task(string desc, string date, bool status = false)

: description(desc), dueDate(date), isCompleted(status) {}

void displayTask() const {

cout << "Task: " << description << ", Due: " << dueDate

<< ", Status: " << (isCompleted ? "Completed" : "Pending") << endl;

}

};

class ToDoList {

private:

vector<Task> tasks;

const string filename = "tasks.txt";

// Save tasks to a file

void saveTasks() {

ofstream file(filename);

if (file.is\_open()) {

for (const auto& task : tasks) {

file << task.description << ";" << task.dueDate << ";" << task.isCompleted << endl;

}

file.close();

}

}

// Load tasks from a file

void loadTasks() {

ifstream file(filename);

if (file.is\_open()) {

string line;

while (getline(file, line)) {

size\_t pos1 = line.find(';');

size\_t pos2 = line.find(';', pos1 + 1);

string desc = line.substr(0, pos1);

string date = line.substr(pos1 + 1, pos2 - pos1 - 1);

bool status = (line.substr(pos2 + 1) == "1");

tasks.emplace\_back(desc, date, status);

}

file.close();

}

}

public:

ToDoList() {

loadTasks(); // Load existing tasks from file when the application starts

}

~ToDoList() {

saveTasks(); // Save tasks to file when the application closes

}

// Add a new task

void addTask(const string& desc, const string& dueDate) {

tasks.emplace\_back(desc, dueDate);

cout << "Task added successfully.\n";

}

// Remove a task by index

void removeTask(int index) {

if (index < 0 || index >= tasks.size()) {

cout << "Invalid index.\n";

return;

}

tasks.erase(tasks.begin() + index);

cout << "Task removed successfully.\n";

}

// Mark a task as completed

void markTaskAsCompleted(int index) {

if (index < 0 || index >= tasks.size()) {

cout << "Invalid index.\n";

return;

}

tasks[index].isCompleted = true;

cout << "Task marked as completed.\n";

}

// View all tasks

void viewTasks() const {

if (tasks.empty()) {

cout << "No tasks available.\n";

return;

}

for (int i = 0; i < tasks.size(); ++i) {

cout << i + 1 << ". ";

tasks[i].displayTask();

}

}

};

void displayMenu() {

cout << "\n--- To-Do List Menu ---\n";

cout << "1. Add Task\n";

cout << "2. View Tasks\n";

cout << "3. Mark Task as Completed\n";

cout << "4. Remove Task\n";

cout << "5. Exit\n";

cout << "Choose an option: ";

}

int main() {

ToDoList toDoList;

int choice;

do {

displayMenu();

cin >> choice;

cin.ignore(); // Ignore leftover newline character from input

switch (choice) {

case 1: {

string description, dueDate;

cout << "Enter task description: ";

getline(cin, description);

cout << "Enter due date: ";

getline(cin, dueDate);

toDoList.addTask(description, dueDate);

break;

}

case 2:

toDoList.viewTasks();

break;

case 3: {

int index;

cout << "Enter task number to mark as completed: ";

cin >> index;

toDoList.markTaskAsCompleted(index - 1);

break;

}

case 4: {

int index;

cout << "Enter task number to remove: ";

cin >> index;

toDoList.removeTask(index - 1);

break;

}

case 5:

cout << "Exiting the application.\n";

break;

default:

cout << "Invalid option. Try again.\n";

}

} while (choice != 5);

return 0;

}