News aggregator

#include <iostream>

#include <vector>

#include <unordered\_map>

#include <string>

#include <ctime>

#include <algorithm>

#include <sstream>

// Define the structure for an Article

struct Article {

    std::string title;

    std::string content;

    std::string source;

    std::time\_t timestamp;

};

// Define the structure for a User

class User {

public:

    std::string username;

    std::unordered\_map<std::string, bool> preferences; // Source -> true/false for preference

    // Add or update a preference for a news source

    void addPreference(const std::string& source, bool value) {

        preferences[source] = value;

    }

    // Display user preferences

    void displayPreferences() const {

        std::cout << "User Preferences for " << username << ":\n";

        for (const auto& pref : preferences) {

            std::cout << "Source: " << pref.first << " - Preferred: " << (pref.second ? "Yes" : "No") << std::endl;

        }

    }

};

// Function to simulate fetching news articles

std::vector<Article> fetchNews() {

    std::vector<Article> articles = {

        {"Breaking News: Market Crash", "The stock market has crashed today due to unforeseen circumstances.", "Source A", std::time(nullptr)},

        {"Tech Update: New Smartphone Released", "The latest smartphone model has been released with impressive features.", "Source B", std::time(nullptr) - 3600},

        {"Sports Roundup: Championship Results", "The results of today's sports championship are in.", "Source A", std::time(nullptr) - 7200},

        {"Entertainment: Celebrity Interview", "An exclusive interview with a popular celebrity.", "Source C", std::time(nullptr) - 14400}

    };

    return articles;

}

// Function to personalize news feed based on user preferences

std::vector<Article> personalizeFeed(const User& user, const std::vector<Article>& articles) {

    std::vector<Article> personalized;

    for (const auto& article : articles) {

        if (user.preferences.find(article.source) != user.preferences.end() &&

            user.preferences.at(article.source)) {

            personalized.push\_back(article);

        }

    }

    // Sort articles by timestamp (most recent first)

    std::sort(personalized.begin(), personalized.end(), [](const Article& a, const Article& b) {

        return a.timestamp > b.timestamp;

    });

    return personalized;

}

// Function to display the news feed

void displayNews(const std::vector<Article>& articles) {

    for (const auto& article : articles) {

        std::cout << "Title: " << article.title << std::endl;

        std::cout << "Source: " << article.source << std::endl;

        std::cout << "Timestamp: " << std::ctime(&article.timestamp);

        std::cout << "Content: " << article.content << std::endl;

        std::cout << "-----------------------" << std::endl;

    }

}

// Function to handle user input for preferences

void handleUserPreferences(User& user) {

    std::string source;

    bool preference;

    std::string input;

    std::cout << "Enter news sources you want to set preferences for (type 'done' to finish):\n";

    while (true) {

        std::cout << "Source Name: ";

        std::getline(std::cin, source);

        if (source == "done") break;

        std::cout << "Preference (1 for Yes, 0 for No): ";

        std::getline(std::cin, input);

        std::stringstream(input) >> preference;

        user.addPreference(source, preference);

    }

}

// Main function

int main() {

    // Create a user and set initial preferences

    User user;

    user.username = "JohnDoe";

    // Display instructions and handle user preferences

    std::cout << "Welcome to the News Aggregator App!\n";

    std::cout << "Set your news source preferences:\n";

    handleUserPreferences(user);

    // Display current preferences

    user.displayPreferences();

    // Fetch news and personalize feed

    std::vector<Article> articles = fetchNews();

    std::vector<Article> personalizedFeed = personalizeFeed(user, articles);

    // Display the personalized news feed

    std::cout << "\nPersonalized News Feed for " << user.username << ":\n";

    displayNews(personalizedFeed);

    return 0;

}