Capstone Project

Team name :- BlackCode (Section - A)

Team Members :-

Dhruv Patel - 202301095

Kavish Patel - 202301074

Yaksh Patel - 202301089

Rishank Dudhat - 202301068

Github link :- Yaksh1311/Capstone (github.com)

Project ID :- P11

Reminder and Task Scheduler

Create an application that manages tasks and schedules using data structures to prioritize and track tasks based on deadlines or importance levels. It should also have reminder functionality as the deadline is approaching.

Introduction

We have created a programme in which users can firstly sign up and can login to access Reminder and Task Scheduler. After they sign up, they can add a task, modify it, delete it and they can see if they have completed it or not. We have added another feature in task addition, User can repeat a task after a finite number of days for a finite number of times. Also, User can change its password.

While adding a task, User will have to provide the following details:

1) Title of the task
2) Task Description
3) Deadline date of the given Task
4) Importance Level (a number between 0 and 100)
5) Answer if User want to Repeat a Task(y/n form)
6) Duration of Repetition(number of days)
7) Frequency of Repetition

For any modification in Task, the user will be asked to choose which part of the task he wants to edit. Also, the user can delete a task. The programme will also show the status of

the task to User. ‘Pending’ status will be shown if the task is incomplete else ‘Completed’ will be shown.

Data Structures Used

❖ Priority Queue(Linked List Implementation):-

The most appropriate data structure for the given problem which was to make a Task Scheduler taking in consideration the deadline and importance of the task was Priority Queue. It was asked to set the priority of the given task on the basis of how close the deadline is and also the importance level which the user assigns the task. So the priority was set by giving more importance to the deadline. Input was taken from the user about both the deadline and the importance level(in percentage), through which a priority was set with 80-20, given to the deadline and importance level respectively.

❖ Vector:-

Vector Data structure is also largely used in the code due to reasons such as efficiency and ease of

implementation. Vector also provides efficient sequential and random access to its elements which is an important property in priority queue because elements are often accessed, removed, and modified according to the priority. Also, vector is a dynamic data structure which makes it better to use than an array.

Pseudocode

(i) FUNCTION: signup

- Parameters passed:- string &username
- Make string variables name, passwd, c_passwd and entry.
- Ask the user to input their name, password and confirmed password, then store it into string variables name, passwd and c_passwd respectively.
- Now in a loop, check if the passwd and c_passwd are the same. If they are not the same, then run the loop again and ask the user to enter the confirmed password again up to 3 attempts. If they are the same, then break from the loop.
- Now open the userlist file(userlist.csv) using fstream.
- Now give the entry variable the value as username,name,passwd.
- Add the entry as an input in the file.
- Close the file stream.
- Time complexity: O(1)

- Space complexity: O(1)

(ii) FUNCTION: login

- Parameters passed:- string &username
- Open the userlist file(userlist.csv) using an input file stream.
- Create a string vector named row.
- Create string variables line, word and pass.
- Make an integer variable count and set its value to 0.
- Read each file from the line using ‘getline’.
- Now make a stringstream named s to split each line into fields using ‘ , ’ and store them in row.
- Now check if the provided username matches the username in the current line.
- If they are not the same, then move on to the next line and check the previous condition again.
- If they are the same, then set count to 1 and proceed to password verification.
- Ask the user to enter their password and store it in the variable pass.
- Now compare the entered password with the password in the file.
- If the password matches, then display “Login successful” and exit the loop.

- If the password doesn’t match, then ask the user to input it again up to 3 attempts.
- If the provided username doesn’t match with any username in the file then display the message “Username does not exist. Please sign up.” and call the signup function.
- Close the input file stream.
- Time complexity: O(n)
- Space complexity: O(1)

(iii) FUNCTION: change_pass

- Parameters passed:- string &username
- Open the userlist file(userlist.csv) using an input file stream.
- Open a new file “userlistnew.csv” for writing using an output file stream.
- Read each file from the line using ‘getline’.
- Now make a stringstream named s to split each line into fields using ‘ , ’ and store them in a vector row.
- Now check if the provided username matches the username in the current line.
- If the username matches, then ask the user to enter their previous password.
- Verify the previous password.
- If the password matches with the password from the file, then ask the user to enter their new password.

- Update the password in the row vector.
- Write the modified and unmodified row to the new file “userlistnew.csv”.
- If the password doesn’t match with the password from the file, then ask the user to enter their previous password again up to 3 attempts.
- Close both input and output file stream.
- Remove the original file “userlist.csv”.
- Rename the new file “userlistnew.csv” to “userlist.csv”.
- Time complexity: O(n)
- Space complexity: O(1)

(iv) FUNCTION: addTask

- Parameters passed:- string username
- Ask the user to input various details about the task, including the task name, description, deadline and importance level(in percentage), then store it in appropriate variables task, description, deadline and Importance_Level respectively.
- It calls a function inputdate to input the deadline date for the task.
- Now check if the input range is within the valid range of 0 to 100. If not, then ask the user to enter a valid importance level.

- Now ask the user whether they want the task to be repeated(y or n).
- If the user chooses to repeat the task(y), ask the user to enter the number of days after which the task should be repeated and the frequency of repetition.
- Store these values in integer variables Duration_days and frequency respectively.
- Create an object t of class Task with the input details: task name, description, deadline, status, importance level, frequency and duration days.
- Open the file “task.csv” for appending. Make an entry string with all the task details separated by commas.
- Write the entry string to the file.
- Close the file.
- Time complexity: O(1)
- Space complexity: O(1)

(v) FUNCTION: deleteTask

- Parameters passed:- int x, string username
- Parameters passed:- int x, string username
- Open the userlist file(task.csv) using an input file stream.
- Open a new file “tasknew.csv” for writing using an output file stream.
- Read each file from the line using ‘getline’.

- Now make a stringstream named s to split each line into fields using ‘ , ’ and store them in a vector row.
- Now check if the provided ID matches the ID in the current line.
- If the IDs don’t match, write the row to new file “tasknew.csv”.
- If the IDs match, set an integer variable count to 1, indicating the task was found and deleted.
- If no matching ID was found(count = 0), display the message “Task Not Found.” and recursively call the function itself to allow the user to try again.
- Close both input and output file stream.
- Remove the original file “task.csv”.
- Rename the new file “tasknew.csv” to “task.csv”.
- Time complexity: O(n)
- Space complexity: O(1)

(vi) FUNCTION: EditFile

- Parameters passed:- int x, string username
- Open the userlist file(task.csv) using an input file stream.
- Open a new file “tasknew.csv” for writing using an output file stream.
- Read each file from the line using ‘getline’.
- Now make a stringstream named s to split each line into fields using ‘ , ’ and store them in vector r.

- Now check if the provided username and ID matches the username and ID in the current line.
- If they match, ask the user to enter the new detail for the specified field and store it in an integer variable newdetail. Then replace the old detail with the new detail in the specified field.
- If they do not match, then display the message “record not found”.
- Close both input and output file stream.
- Remove the original file “task.csv”.
- Rename the new file “tasknew.csv” to “task.csv”.
- Time complexity: O(n)
- Space complexity: O(1)

(vii) FUNCTION: edit

- Parameters passed:- string &username
- Display a menu with options for what aspects of the task to edit: task, description, deadline, importance level, day cycle frequency or frequency of repetition.
- Now make an integer variable index.
- Use switch case statements to read the user's choice to determine which choice the user selected.

- Depending on the user's choice, call the other function 'EditFile', passing it the index corresponding to the user's choice and the username.
- If the user enters invalid choice, display a message "Wrong choice. Please enter again." and recursively call the edit function.
- Time complexity: O(n)
- Space complexity: O(1)

(viii) FUNCTION: completeTask

- Parameters passed:- string username
- Open the userlist file(task.csv) using an input file stream.
- Open a new file "tasknew.csv" for writing using an output file stream.
- Read each file from the line using 'getline'.
- Now make a stringstream named s to split each line into fields using ' , ' and store them in vector r.
- Now check if the provided username and ID matches the username and ID in the current line.
- If they match, update the status of the task to 'Completed'.
- If a task has repetitions, decrement the frequency of repetitions and adjust the deadline according to the day cycle frequency of the task.
- Write the modified and unmodified row to the new file.
- Close both input and output file stream.

- Remove the original file “task.csv”.
- Rename the new file “tasknew.csv” to “task.csv”.
- Time complexity: O(n)
- Space complexity: O(1)

(ix) FUNCTION: isLeapYear

- Parameters passed:- int year
- Check if the provided year is a leap year or not.
- Time complexity: O(1)
- Space complexity: O(1)

(x) FUNCTION: daysinMonth

- Parameters passed:- int m, int y
- Initialize an array daysInMonths containing the number of days in each month of a non-leap year.
- Retrieve the number of days for the given month m from the array. Access the array using the index ‘m-1’ because the array is zero-based indexed.
- If the given month is February (m == 2) and the given year y is a leap year (determined using the isLeapYear function), it updates the number of days to 29.
- Time complexity: O(1)
- Space complexity: O(1)