Software Design Description

for

TODO: Task Management System

CS 471

Draft 0.1.2

Prepared by:
Blake Eggemeyer

December 2, 2011

Contents

1 Introduction

1.1 Overview

This document is intended to be used by the developer of the TODO software.

1.2 Stakeholders

The stakeholder in the design is also the client.

1.3 Definitions

1.3.0.1 Should: Requirements with this marker are desired, but not crucial, and will be a part of the final deliverable contingent on time and progress.

1.3.0.2 TBD: Acronym for To Be Determined. This is used in this document to signify that the information necessary for a part of this document is “To Be Determined”.

1.3.0.3 TODO: Working name of the project.

1.3.0.4 User: The person, or persons, who operate or interact directly with the product.

1.3.0.5 Will: Requirements with this marker are guaranteed to be in the final delivered product.

1.3.0.6 Item: Item refers to a data object. This object is either an appointment or a task.

1.3.0.7 CSV: CSV is an acronym for Comma Separated Value. This is a standard and common format of simple tabular data.

1.4 References

The `1998 - IEEE Standard for Information Technology - Systems Design - Software Design Descriptions` was referenced to produce this document.

1.5 Revision tracking

0.0.1	Nov 11	Empty document created.
0.0.2	Nov 17	Framework added.
0.0.3	Nov 18	Framework extended.
0.0.4	Nov 19	Design.
0.1.0	Nov 21	Version that will be turned in.
0.1.1	Nov 30	Making alterations for resubmission.
0.1.2	Dec 2	Version resubmitted.

2 Design Considerations

2.1 Programming Languages

TODO will be implemented in C++ due to the programmers experience with that language.

2.2 Project Management

This project will use `Git` version control in conjunction with `GitHub` to keep track of changes. The repository can be reached at `https://github.com/blake6489/TODO-TaskManager-/`.

3 Data Storage

3.1 Data Dictionary

3.1.1 Configuration

TODO will reference a configuration file by the name of `config` in the same directory as the executable.

3.1.1.1 Update frequency: The active list will be updated at the frequency in seconds specified by the user. The system will not support updates more frequent than 1 second. The default value is 15 minutes.

3.1.1.2 Active list length: This value will determine what portion of the active list is printed to the terminal. This value does not limit the length of the list, only which entries are displayed. The default value is 15 items.

3.1.2 Task

3.1.2.1 Unique ID: Non user editable `int`. This is the index value used for internal reference to the task.

3.1.2.2 Name: Data type `string`.

3.1.2.3 Description: Data type `string`.

3.1.2.4 Project: Data type `string`.

3.1.2.5 Due date: Input as three integers. Stored as Unix epoch time `int`.

3.1.2.6 Time estimate: Input as three integers. Stored as Unix epoch time `int`.

3.1.2.7 Elapsed time: Calculated based on 'Working on top item in list' function. Stored as Unix epoch time `int`.

3.1.2.8 Priority: Integer representing number of tasks from the top.

3.1.2.9 Prerequisites: Integer representing the unique Id of another task.

3.1.3 Appointment

3.1.3.1 Unique ID: Non user editable `int`. This is the index value used for internal reference to the appointment.

3.1.3.2 Name: Data type `string`.

3.1.3.3 Description: Data type `string`.

3.1.3.4 Project: Data type `string`.

3.1.3.5 Date: Input as three integers. Stored as Unix epoch time `int`.

3.1.3.6 Estimated duration: Input as three integers. Stored as Unix epoch time `int`.

3.1.3.7 Time worked: Calculated based on ‘Working on top item in list’ function. Stored as Unix epoch time `int`.

3.1.3.8 Priority: Integer representing number of tasks from the top. This is re-calculated every minute to move the appointment up the active list.

3.1.4 Active list

The active list will be the list of all active items. The ordering of this list will represent the priority of the items in it with the items in the zeroth position being the most important.

3.1.5 Inactive list

The Inactive list will be the entirety of all completed items. As the inactive list grows in size performance may suffer.

4 Actions

4.1 Terminal

4.1.0.1 New item: By running `new` the user will add new items to the list and enter the necessary data fields.

4.1.0.2 Working on top item in list: By running `top` the user will initiate the accumulation of time worked on the top item in the list. If there are no items in the list, the program will return test to the command shell stating `No items in list`.

4.1.0.3 Mark inactive: By running `inactivate [Unique ID]` the specified task will be marked inactive and removed from view.

4.1.0.4 Completed: By running `complete` the top task in the list will be marked as inactive and completed, and the time worked will stop accumulating. The item will be moved to the Inactive list.

4.1.0.5 Done for now: By running `stop` the top task in the list will stop accumulating time worked.

4.1.0.6 Changing list order: By running `move [Unique ID] [new priority]` the specified task will be moved to the specified priority position in the list, unless the prerequisites for that task are ahead of it. In this case the option will be given to move all prerequisites of this task to the specified position.

4.1.0.7 Modify a user created field: By running `modify [Unique ID] [option]` the specified task will be modified, with the given piece of data being changed to the given value.

4.1.0.8 Make item top priority: By running `move [Unique ID] 0` he specified task will be moved to the top priority position in the list, unless the prerequisites for that task are ahead of it. In this case the option will be given to move all prerequisites of this task to the specified position.

4.1.0.9 Delete: There is no deletion functionality

4.1.0.10 View complete list: By entering `all` the user will be able to see the entire complete list in the terminal.

4.1.0.11 Export complete list: By entering `export` the user will be able to export the entire complete list to a `CSV` file.

4.2 Automatic actions

4.2.1 Sorting

The list the user sees will be sorted using th following rules.

4.2.1.1 Priority at creation time: When a task is added the user may provide a priority for that task. This number indicates how many elements below the top element this item should be. If this number is larger than the current number of elements the task is pushed as the last item in the list. After this the priority has no meaning as other items may be inserted or removed before it.

4.2.1.2 Appointment position in list:

4.2.1.2.1 When the active list is called to be shown the tasks will be copied into a temporary list and maintain their positions in the list relative to each other.

4.2.1.2.2 The appointments will be inserted into the temporary list.

4.2.1.2.3 The first appointment inserted will be the one farthest in the future.

4.2.1.2.4 The position in the active list will be calculated by time till appointment divided by user defined update interval.

4.2.1.2.5 If the calculated position is larger than the entire list, the appointment will be pushed onto the end of the list.

5 Views

5.1 Viewing Active List

The user will see the list of tasks and appointments to be done in order of priority. These tasks will be those that are not marked inactive, or completed. The number of items shown is specified by the user in the `config` file and defaults to 15 items.

```
Unique ID, Appointment Y/N,       Name,      Description,      Project,                Due Date, Estemated Time Remaining, Time Worked, Prerequesites
        4,                1, namething2, descprojectthing, projectthing, Tue Nov 29 19:51:57 2011,                5 hours,     2 hours,          none
        5,                1, namething4, descprojectthing, projectthing, Tue Nov 29 19:56:57 2011,               52 hours,   2.5 hours,          none
        6,                1, namething7, descprojectthing, projectthing, Tue Nov 29 19:58:37 2011,                6 hours,     2 hours,          none
        2,                0, namething6, descprojectthing, projectthing, Tue Dec  6 00:00:00 2011,              3.5 hours,     2 hours,          none
        7,                0, namething5, descprojectthing, projectthing, Sun Dec  4 00:00:00 2011,              100 hours,   0.5 hours,          none
        1,                0, namething1, descprojectthing, projectthing, Tue Nov 29 05:00:00 2011,               12 hours,     2 hours,          none
        3,                0, namething3, descprojectthing, projectthing, Thu Dec  1 00:00:00 2011,                3 hours,     2 hours,          none
```

Figure 1: Example table layout

5.2 Viewing All

The user will see the entire list of work done on items regardless of completion status when executing the `all` or `export` functions. This view will match the active items table above, but with a few additional columns.

6 Interface

6.1 Terminal

Using a command line interface would allow the TODO software to be used in an open Linux terminal. This interface option is for more limiting for typical users than a GUI but will be simpler to implement and will achieve the goals. The terminal would print out the active list in order of which priority and then leave a line at the bottom for the user to enter the commands listing the Actions section.

6.2 Glade2

Glade2 is a possible user interface design tool that would allow TODO to be implemented in a C++ GUI.

7 Test Cases

A set of automatic tests will be written to perform inputs on the system and the outputs will be checked by a human. The state that the system should finish each test in is known but not easily testable. Most of the tests will be to ensure that improper actions are not executed and that the data is not corrupted.

8 Traceability matrix

	Functional												Non-functional			
	3.1.1	3.1.2	3.1.3	3.1.4	3.1.5.1	3.1.5.2	3.1.6.1	3.1.6.2	3.1.6.3	3.1.6.4	3.1.6.5	3.1.7.1	3.2.0.1	3.2.0.2	3.2.0.3	3.2.0.4
3.1.1.1													✓			
3.1.2	✓															
3.1.3		✓														
3.1.4			✓													
3.1.5				✓										✓		
4.1.0.1																
4.1.0.2							✓									
4.1.0.3								✓								
4.1.0.4									✓							
4.1.0.5										✓						
4.1.0.6						✓					✓					
4.1.0.7					✓											
4.1.0.8						✓										
4.2.1.2												✓	✓			
4.1.0.9																✓
4.1.0.10															✓	
4.1.0.11															✓	
5.2															✓	