Software Design Description

for

TODO: Task Management System

CS 471

Draft 0.1.0

Prepared by:
Blake Eggemeyer

November 21, 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.

1.4 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.

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.

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.

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 recalculated form the `complete list` each time a recalculate is requested. This will occur when changes are made to the items in the list and at scheduled intervals.

3.1.5 Complete list

The complete list will be the entirety of all items regardless of there completion status. This list will be only list which is stored. All other lists will be calculated from the complete list. As the complete list grows in size performance may suffer.

4 Actions

4.1 Command line

4.1.0.1 Working on top item in list: By running `todo 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`.

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

4.1.0.3 Completed: By running `todo complete` the top task in the list will be marked as inactive and completed, and the time worked will stop accumulating.

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

4.1.0.5 Changing list order: By running `todo 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.

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

4.1.0.7 Make item top priority: By running `todo 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.

4.1.0.8 Delete: There is no deletion functionality

4.2 Automatic actions

4.2.0.1 Appointment The position of an appointment will change at an interval set by the user so that the appointment appears at the top of the list before it actually occurs. The position in the active list will be calculated by time till appointment divided by user defined update interval. With the soonest appointment calculated first.

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.

5.2 Viewing All

The user will see the entire list of work done on tasks regardless of completion status.

6 Interface

6.1 Command line

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.

6.2 Glade2

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

7 Test Cases

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						✓										
4.2.0.1												✓	✓			
4.1.0.8																✓
5.2															✓	