Version Control

[Project Name]
Requirements Traceability Matrix

Submitted to: Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by: [List of Members in Alphabetical Order]

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering
1st Sem AY 2014-2015

History Revision:

Revision Date	Person Responsible	Version Number

Version Control

Modification

RTM ID

Use-Case 1.0 View To-do List

Use-Case 2.0 Edit Cell

Use-Case 3.0 Add Task to List

Use-Case 4.0 Delete Task from List

Use-Case 5.0 Set Time

Use-Case 6.0 Create Pie Schedule

Use-Case 6.1 Add Task/Slice

Use-Case 6.2 Arrange Task/Slice

Use-Case 6.3 Resize Slices

Use-Case 6.4 Delete Task/Slice

Use-Case 7.0 Check Task

Use-Case 8.0 Reset

Use-Case 9.0 Start

Use-Case 10.0 Respond to Alarm

Use-Case 11.0 Adjust Time

Use-Case 12.0 Pause/Unpause

Description	Scenarios
The list stores the tasks listed by the user. View To-do List shows a column of tasks and a column of dates. Each row contains a task name and corresponding schedule.	S1: Student views the To-do List
The list is a table with two columns and a definite number of rows. It can be edited by editing its cells one at a time. The user can change the tasks' names and dates. After a cell is modified, the list is resorted.	S1: Student edits a task name cell S2: Student edits a task date cell
A task is added by inputting its name/label. The user may or may not input a scheduled date. A new row containing the entry is inserted into the list based on how the list is currently sorted.	S1: Student adds a task to the list
The user can choose to remove a task from the list. A task is deleted by deleting the cell containing the task name. This removes the entire row from the list. If a task in the list is added to the pie, once it is done, the task is automatically deleted from the list.	S1: Student deletes a task from list
The time frame of the pie is set by the user. The input can be either the duration or the start and end times expected to accomplish all the tasks in the schedule. Once the user has started the timer of the pie, the application will run during the given time.	S1: Student sets the time of the new pie
This is the main operation of the application. The pie schedule is manually created by the user by adding, arranging, allotting time, and deleting. This is linked to the To-do List.	S1: Student creates pie schedule
A task is added by inputting its name and choosing a color. It is added to the pie schedule as a slice with the chosen color. The user can add up to 18 tasks only. The user can also select tasks from the To-do list. The list will show only the tasks scheduled on the same date the user creates the pie.	S1: Student adds a task from list S2: Student adds a task by inputting
The tasks can be easily rearranged. The slices are moved by dragging them clockwise or counterclockwise along the pie. This can be done when creating the pie schedule and after Start when paused. After Start, finished tasks cannot be rearranged.	S1: Student moves a task when creating a pie S2: Student moves a task after Start when paused
Only after all the tasks have been added when creating the pie and when timer is paused can the user change their sizes. Once resizing has occurred, adding a task will be impossible. Slices are resized by dragging their boundary lines. The size of a slice is used to compute for its corresponding task's allotted time. The ratio of a task's duration to the time frame of the whole pie is equal to the ratio of the slice's area to the pie's.	S1: Student resizes a slice of pie when creating a pie S2: Student resizes a slice of pie when paused after Start

Tasks can be deleted one at a time. Since there is no edit task function, the alternative is to delete and add. When a task is deleted, the pie adjusts such that the area previously occupied by the deleted slice is distributed equally among the slices.	S1: Student deletes a slice of pie
The user can view the detail of a slice by clicking it. This will show the task and its remaining allotted time. This can be done anytime.	S1: Student views a task in the pie
This deletes all the tasks resulting to an empty circle. It is useful when the user wants to add tasks after resizing. This can be done anytime.	S1: Student resets the pie schedule
This starts the timer of the pie. Also, the user picks from which task to start. Once the start button is pressed, the pie schedule can only be resized and rearranged during pause.	S1: Student starts the pie
The application will alarm once for a few seconds whenever a task runs out of time. At the same time, it will ask the user if the task is finished or not. If yes, the next task will be shown and the timer will resume. Otherwise, the user will have the option to auto-adjust.	S1: Student responds to an alarm when the task is not finished
	S2: Student responds to an alarm when the task is finished
When the time allotted for a task runs out and the user is not yet done, he/she has the option to let the application redistribute the remaining time among the unfinished tasks. If the user chooses to do so, the Adjust Time will recalculate the durations.	S1: Student uses auto-adjust
The Pause Timer function enables the user to pause the timer for the schedule. Its purpose is to make it convenient for the user when necessary to tend to things not included in the pie. The Unpause Timer function resumes the timer. In setting the time, if the input was start and end times instead of duration, the user will be given two options: move end time or recalculate the durations for the unfinished tasks. Selecting move end time changes the end time by adding the duration of the pause. While choosing recalculate the durations results to Adjust Time.	S1: Student pauses timer
	S2: Student unpauses timer, chooses to move end time
	S3: Student unpauses timer, chooses to adjust durations

Priority Status	Has Prototype?	Boundary Class	Require
Should Have	Yes	ListUI.java	
Should Have	No	ListUI.java	
Should Have	No	ListUI.java	
Should Have	Yes	ListUI.java	
Should Have	No	ListUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	
Must Have	No	PieScheduleUI.java	

Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	Yes	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java
Must Have	No	PieScheduleUI.java

ements Phase			Design Phase
Control Class	**Entity Class**	**UI Design**	**Controller Design**
ListController.java	List.java		
ListController.java	List.java		
ListController.java	List.java		
ListController.java	List.java		
ListController.java	List.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		
PieScheduleController.java	PieSchedule.java		

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

PieScheduleController.java PieSchedule.java

Data Design