Sarah Saad

14 January 2018

CSC 4556

Documentation

**Functional requirements:**

1. The user should be able to add tasks to the to-do list
2. The user should be able to delete tasks to the to-do list
3. The user should be able to filter the list by a specific status
4. The system should generate the entire to do-list upon request
5. The system should allow updates to the status of the list by the user
6. The system shall show how many tasks have a specific status
7. The system should show the total number of tasks

**Non-functional requirements:**

1. The system should not take more than 5 minutes to load a reasonably sized To-do list
2. The application should not crash when displaying to-do list
3. The application should not crash when adding, deleting or changing the status of tasks
4. The application should make use of the WAMP server
5. The system should be available on a daily basis
6. Users should be able to use the application with no more then 15 minutes of instruction
7. Users should be able to use the application at no cost

**System Architecture Diagram**

**Computer**

**LocalHost (Server)**

**Hardware**

**Project Directory**

**PHP files**



**User**

Interacts with Keyboard and mouse

**Phpadmin (Database)**

**Table (To-Do list)**

**User Interface**

**Data Flow Diagram (Level 2)**

Incentive to complete Task

User

Complete Task

Given Task

**Use cases:**

|  |
| --- |
| **Name:** Viewing Tasks |
| **Brief Description:** This user is unaware of tasks he or she must complete. This indidual would like to see a list of said tasks. |
| **Actors:** Hardworking individual |
| **Preconditions:** They must have entered tasks in their list prior |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on view list link 3. User looks at list 4. User is now informed of what he or she needs to do at some date |
| **Alternative Flows:** none |
| **Exception Flows:** none |
| **Post Conditions:** User is now aware of the tasks he or she needs to accomplish |

|  |
| --- |
| **Name:** Adding a single Tasks |
| **Brief Description:** The userhas been given a task he or she needs to remember. The individual wants to add this task to their To-Do list |
| **Actors:** Hardworking individual |
| **Preconditions:** none |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on “add task button” 3. User fills out form indicating the task 4. User fills out form writing down the date the task should be done |
| **Alternative Flows:** none |
| **Exception Flows:** none |
| **Post Conditions:** Database is now updated |

|  |
| --- |
| **Name:** Adding a multiple Tasks |
| **Brief Description:** The userhas been given multiple tasks he or she needs to remember. The individual wants to add all of them to their To-Do list |
| **Actors:** Hardworking individual |
| **Preconditions:** none |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on “add task button” 3. User fills out form indicating the task 4. User fills out form writing down the date the task should be done 5. User uses the provided return to homepage button 6. User repeats steps 2-5 as needed |
| **Alternative Flows:** none |
| **Exception Flows:** none |
| **Post Conditions:** Database is now updated, tasks were removed |

|  |
| --- |
| **Name:** Deleting a single Tasks |
| **Brief Description:** The userno longer has to do a task he or she was assigned and would like to remove it. The user could also simply wants to delete a completed task. |
| **Actors:** Hardworking individual |
| **Preconditions:** There have to be tasks placed into the database prior |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on “deletes task button” 3. User fills out form indicating the task 4. User submits and deletes task |
| **Alternative Flows:**  3a. User forgets which task to delete  3b. User clicks view list on page |
| **Exception Flows:** none |
| **Post Conditions:** Database is now updated |

|  |
| --- |
| **Name:** Deleting Multiple Tasks |
| **Brief Description:** The userhas multiple tasks he or she no longer has to complete |
| **Actors:** Hardworking individual |
| **Preconditions:** none |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on “delete task button” 3. User fills out form indicating the task 4. User submits and task is removed from database 5. User uses the provided return to homepage button 6. User repeats steps 2-5 as needed |
| **Alternative Flows:** none |
| **Exception Flows:** none |
| **Post Conditions:** Database is now updated |

|  |
| --- |
| **Name:** Changing Task Status |
| **Brief Description:** The userhas completed a task. The user would now like to update the task on the list to reflect this. |
| **Actors:** Hardworking individual |
| **Preconditions:** The task must already be in database |
| **Basic Flow:**   1. User gets on homepage 2. User clicks on “completed task” button 3. User fills out form indicating the task 4. User fills out date of task 5. User submits |
| **Alternative Flows:** none |
| **Exception Flows:** none |
| **Post Conditions:** Database is now updated, task status is changed |

**Sequence Diagram:**

1. Adding a task

Database

Server

UI

User

User Clicks on add Task button on web page

2. Deleting a Task

User

UI

Server

Database

3. Viewing To-Do List

User

UI

Server

Database

4. Updating Task Status

User

UI

Server

Database

Database Design one for the system, 3+ tables

Class diagram one for the system, 3+ classes

Test Case 6 operations to be tested