**“ToDo” Application** Keith Zane – FS9346

## Project Objective:

The goal of this project is to provide an easy to use ToDo list application available through a webpage interface. It should have an html front end and a mySQL backend using php for communication between the two.

## Functional Requirements:

|  |  |
| --- | --- |
| Requirement Identification | Requirement Description |
| FunctReq.01 | The user shall be able to add a task to the list of available tasks by clicking on an “add task” button.   * When the user clicks on the button they should be directed to another page to fill out information for the task * This new window should also have the ability to save the task or return without saving. |
| FunctReq.01.01 | When filling out the task information, the user should be prompted with required information before saving. |
| FunctReq.02 | The user shall be able to view the task that they created apart from the task list by clicking on a “view task” button.   * When the use clicks on the button they should be directed to a page that looks similar to the add task window. * The user should be able to update that task or return to the main task list |
| FunctReq.02.01 | When editing the task information for an already created an account, the user should be prompted with information that is not valid and needs to be changed. |
| FunctReq.03 | The user shall be able to remove a task from the list of tasks by clicking in a “delete task” button.   * There should be a prompt that pops up asking whether they are sure they want to delete this task * This prompt should give the user the option of deleting the task, archiving the task instead, or canceling the operation. |

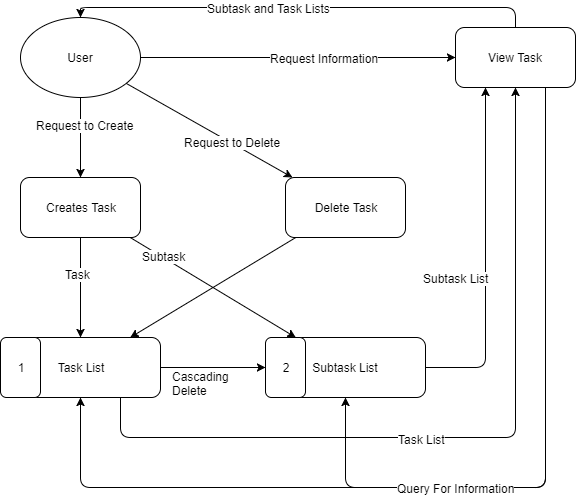
## Non-functional requirements:

|  |  |
| --- | --- |
| Requirement Identification | Requirement Description |
| NonFunctReq.01 | If the user is on the “ADD TASK” screen and wishes to go back to the previous screen without adding a task, a prompt should appear warning the user that the information will not be saved. This is to help prevent accidental loss of information. |
| NonFunctReq.02 | If the user chooses to delete a task, a confirmation window should pop up to ensure that the user does in fact want to permanently remove a task. |
| NonFunctReq.03 | Each window must conserve the amount of space that the content takes on each window by implementing a wrapper class. |
| NonFunctReq.04 | The task list should have different sections, one for all the major tasks and another section for tasks that have been archived. |

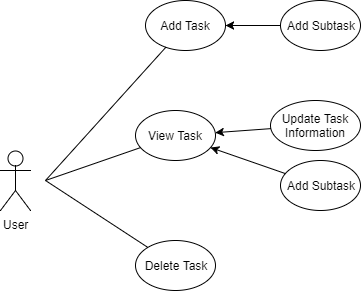
## System Architecture Diagram:



## Data Flow Diagram:



## Use cases:

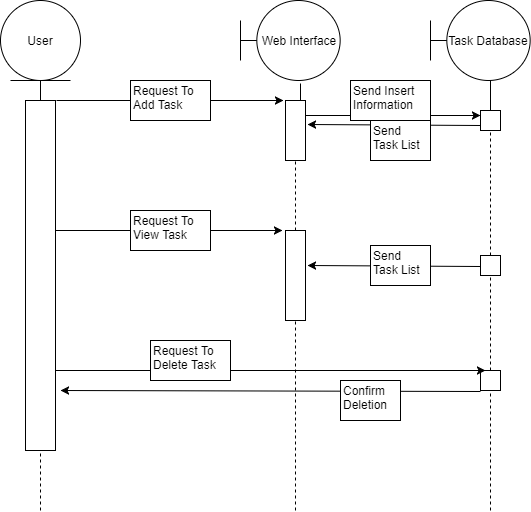


|  |  |
| --- | --- |
| Use Case Number: | 1 |
| Use Case Name: | User Adds A Task |
| Description: | A user has a task they wish to add to the task list. They will create a task and give it the amount of information they deem appropriate for the task. If they need they can also create additional subtasks to help track the task. |

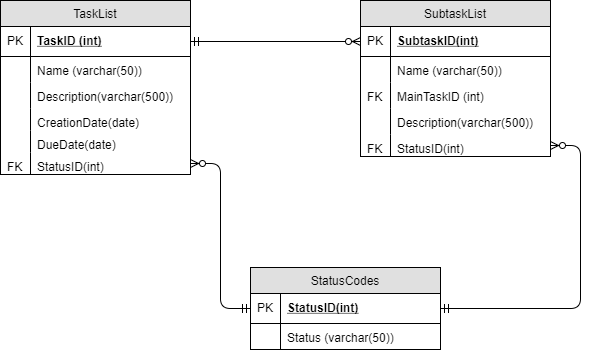
|  |  |
| --- | --- |
| Use Case Number: | 2 |
| Use Case Name: | User Views A Task |
| Description: | A User wants to see the information about the task that they created. If they need they can change the information on that task. They can also change the information on the subtasks and create more if they need it. |

|  |  |
| --- | --- |
| Use Case Number: | 3 |
| Use Case Name: | User Deletes A Task |
| Description: | A User no longer wants to see a task in the task list. They will delete the task and it shall be removed from the task list. This is nonreversible. |

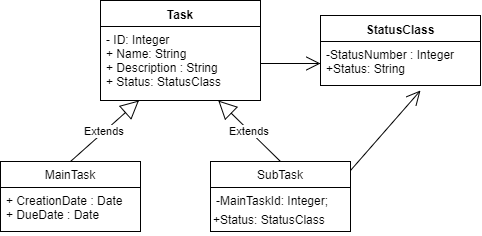
## Sequence Diagram:



## Database Design:



## Class Diagram:



## Test Cases:

|  |
| --- |
| **TestCase01** – AddTask Page – Add a task with information to the task list  A user should be able to add a task as well as any additional information to the task list at any time.  Precondition: None  Test Steps:   1. Click “Add Task” button on the index.php (Main Landing Page) 2. In the “Name” field, fill out the name of the task 3. In the other fields fill out any additional information 4. Click “Add Task” button on the addnewtask.php (Add Task View Page)   Expected Result: A task with the name provided should appear on the main page. |

|  |
| --- |
| **TestCase02** – ViewTask Page – Update task with new information  A user should be able to change the information of an already created task, including changing its name, description, status, and add subtasks  Precondition: The task that the user wishes to change must exist  Test Steps:   1. Click “View Task” button on the index.php (Main Landing Page) for the corresponding task 2. Change any of the information as needed to update the task 3. Click “Update Task” button on the viewtask.php (Task View Page)   Expected Result: When the user views the task again, the task should have the new information that the user put in. |

|  |
| --- |
| **TestCase03** – Index Page – Remove an already existing task  A user should be able to delete any task that they wish  Precondition: The task that the user wishes to delete must exist  Test Steps:   1. Click “Delete Task” button on the index.php (Main Landing Page) for the corresponding task 2. Click “Ok” on the dialog that pops up confirming deletion.   Expected Result: The screen should refresh with the task no longer in the list. |