TODO-List Application

By Matthew Weberman

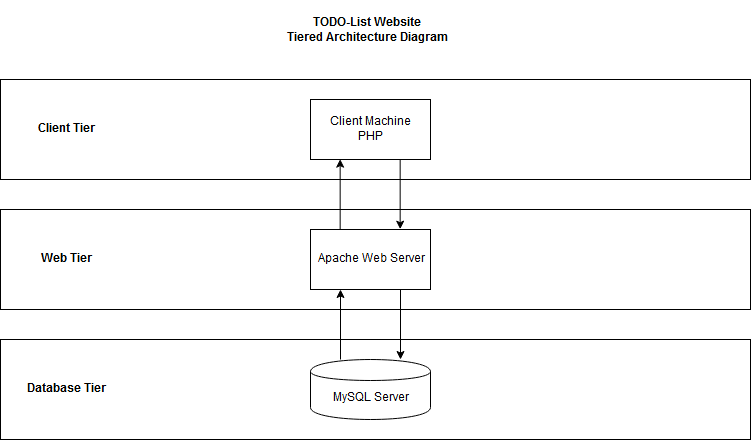
Functional Requirements:

* Insert user-inputted string into the TODO-list
* Remove user-specified string from the TODO-list
* Display TODO-list to user

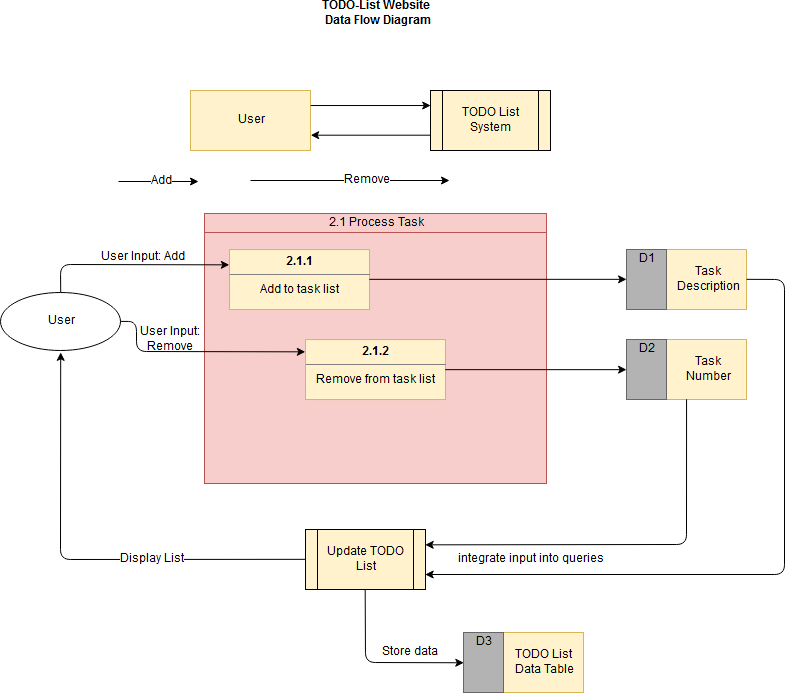
Non-Functional Requirements:

* Updated table shall be displayed on each load of the page/insert function/remove function
* Strings are limited to 70 characters in length
* “Very large” limit on number of strings stored

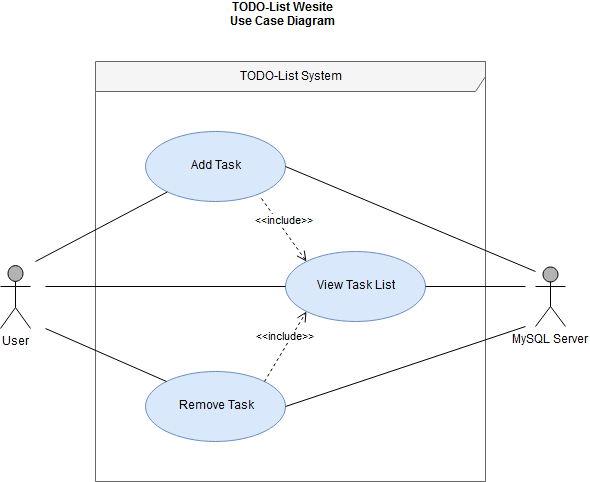
System Architecture Diagram:



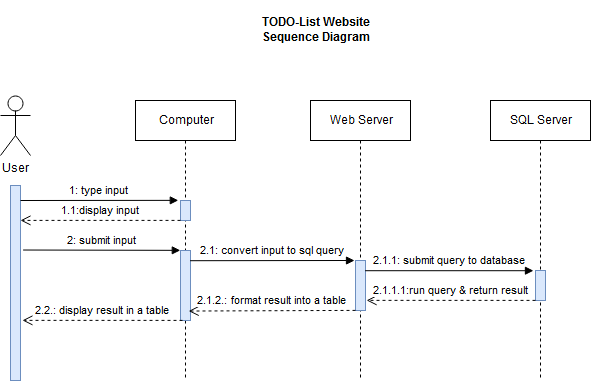
Data Flow Diagram:



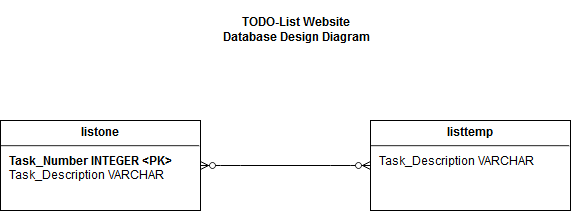
Use Case:



Sequence Diagram:



Database Design:



Class Diagram:

Test Case:

|  |  |
| --- | --- |
| **Test Scenario** | Check TODO-List functoinality |
| **Test Case** | Check adding multiple values and removing multiple values |
| **Pre Conditions** | MySQL Server must be installed with no root password, otherwise password must be typed into index.php inside the $password variable and saved |
| **Test Step** | 1. Launch website  2. Type task into the text box next to the “Submit” button  3. Press submit button to add string into table  Repeat steps 2 and 3 until at least 5 different tasks are in the list  4. Enter value for a middle task into the text box next to the “Remove” button.  5. Click the remove button  6. Repeat steps 4 and 5 with the first and last numbered tasks in the list |
| **Test Data** | Submit #1: one  Submit #2: two  Submit #3: three  Submit #4: four  Submit #5: five  Remove #1: 3  Remove #2: 1  Remove #3: 3 |
| **Expected Result** | Remaining in the table should be 2 tasks:   |  |  | | --- | --- | | **Task\_Number** | **Task\_Description** | | 1 | two | | 2 | four | |
| **Actual Resuls** | |  |  | | --- | --- | | **Task\_Number** | **Task\_Description** | | 1 | two | | 2 | four |   Received expected output |
| **Pass/Fail** | Pass |

Workflow Breakdown

|  |  |  |
| --- | --- | --- |
| **Task/Notes** | Estimated Time | Actual Time |
| Learn how to do parts of the documentation  Notes: Need to look up each part just to understand how they are done. | 2 hours | Start: 9:35AM – 10:32 AM  Time: 57 minutes |
| Create Repository | 0.5 hours | Start:10:32AM – 10:38AM  Time:6 minutes |
| Install WAMP on Laptop and Desktop / “Hello World” | 2 hour | Start: 10:38AM – 10:44AM  11:10AM – 11:28AM  11:58AM – 12:08AM  Time: 44 minutes |
| Test MySQL with the webpage | 1 hour | Start: 12:30PM – 12:40PM  Time: 10 minutes |
| Create code | 4 hours | Start:12:40PM – 12:50PM  2:50PM – 3:26PM  8:34PM – 9:05PM  9:20PM – 1:20AM  Time: 5 hours 15 minutes |
| Clean code / make UI (if time) | 2 hours | Start:  End:  Time: |
| Create Documentation: |  |  |
| * Functional requirements (required/desired) | 5 minutes | Start:2:23PM – 2:28PM  Time:5 minutes |
| Non-functional Requirements | 20 minutes | Start:2:32:PM – 2:37PM  Time:5 minutes |
| * System Architecture Diagram | 1 hour | Start:1:25AM – 1:35AM  12:57PM – 1:27PM  Time:40 minutes |
| * Data Flow Diagram | 1 hour | Start: 1:27PM – 2:20PM  Time: 53 minutes |
| * Use Case | 1 hour | Start:2:22PM – 2:41PM  Time:19 minutes |
| * Sequence Diagram | 1 hour | Start:5:31 PM – 6:02PM  Time:31 minutes |
| * Database Design/E-RDiagram | 30 minutes | Start:6:18 – 6:38  End:  Time:20 minutes |
| * Class diagram | 1 hour | Start:  End:  Time: |
| * Test Case | 2 hours | Start:6:39PM – 7:19PM  Time:40 minutes |
|  | Est Total Time: 17.417 hours | Actual Total Time: 645 minutes/10.75 hours |

Drawback notes:

* Had to learn most of the documentation from scratch
* Had trouble with code:
  + stack functionality not working correctly (fixed by using a temporary table so the main table can be dropped and remade to reset the primary key’s auto\_increment value).
  + Found a way around having every GET variable show up in the url for a cleaner address bar.
  + Found a way to have the page reload on each input submission so the table can be displayed in “real-time.”
* Unsure of how to divide program into separate classes to create the class diagram