TODO Application Documentation

# Functional Requirements

The To Do web application has three functional requirements.

* Add: After entering the data in the text fields, the add function will insert the data into the database.
* Delete: After entering the data in the text fields, the delete function will remove the information in the text field from the database.
* View: This function will allow the user to see all data stored in the database in a form of a table on the view page

# Non-Functional Requirements

The web application has four non-functional requirements.

* Priority Level: The data entered for priority level must be a minimum of 1 and a maximum of 3. This ensures data displayed will be organized by priority level, which is high (1) to low (3).
* Connection: Current connection requirements are to use xampp and phpmyadmin. These two services will create a connection between the server language which is PHP and the database which is MySQL.

The following two non-functional requirements have not been implemented yet.

* Login: Each person has a unique username and will not be case-sensitive.
* Password: Passwords must be 8-15 characters long and will be case-sensitive.

# System Architecture Diagram

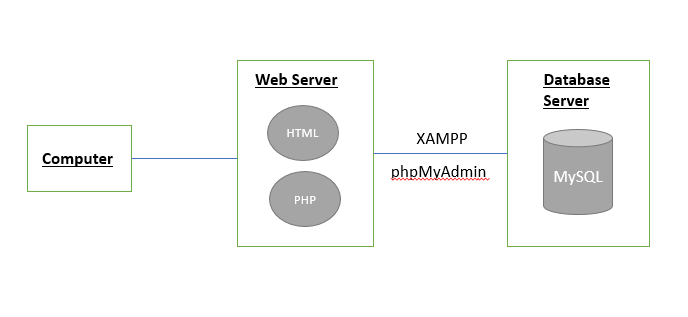


Figure 1: To Do Application System Architecture Diagram

The diagram above portrays the different components of the system. These are the main portions which allow clients to interact with a server.

# Data Flow Diagram

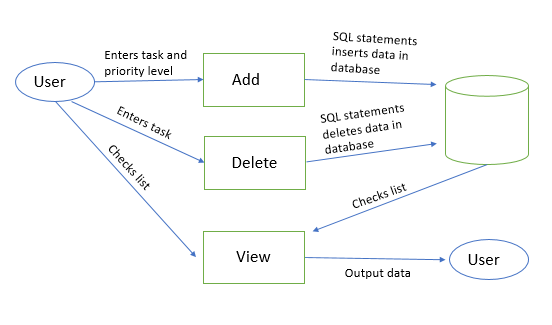


Figure 2: Data Flow Diagram

The diagram shows the input and output of each functions. It shows the flow of data beginning from the user’s initial use of the web application. If the user enters data into the text field regarding the task name and priority level, the ‘Add’ function will insert the data into the database. This insertion occurs through SQL statements which stores it into the MySQL database. If the user decides to delete a task, they will enter data into the text field. The ‘Delete’ function will then use SQL statements to remove the item from the database. In both, Add and Delete, the user has options to view the updated database. To check the list, it will call the ‘View’ function which outputs the data onto the screen for the user to view.

# Use Case

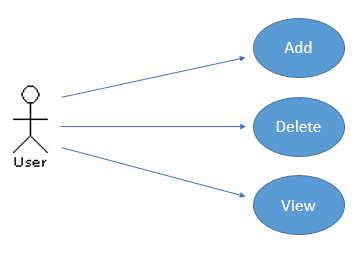


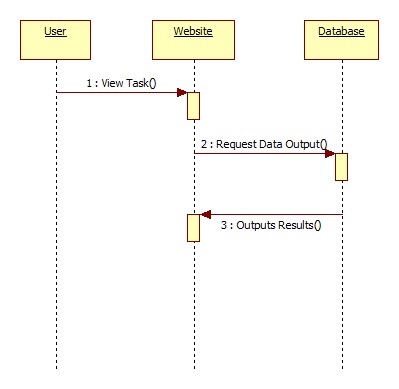
Figure 3: Use Case

The website has three forms of usage available for users who would like to use it. The web application can be used to add tasks to the To Do list. These tasks can be added with either a simple or descriptive task, depending on the user’s preference. Users can also view the list through the website and check what tasks still need to be completed. Another usage available is to delete tasks that are no longer needed or have been completed.

# Sequence Diagram

(b)

(a)



(c)

Figure 4: Sequence Diagrams with adding (a), deleting(b), and viewing(c).

Sequence diagrams show the process of interaction between the user and the systems. The first two diagrams, (a) and (b), have a similar process. In both situations, the user sends a request to the website to either add or delete a task. The website then sends that request to the database in forms of SQL statements. This SQL statement makes the necessary changes and updates the database. The database then outputs the results back to the website.

The sequence diagram for viewing a task is slightly different from the others. In this diagram, the user sends a request to the website to view the data. The website transfers that request to the website. This transfer occurs in forms of SQL statements in here as well. The statement would be:

SELECT \* FROM viewtask ORDER BY Priority ASC;

The database then prints the data on the website based on this SQL statement. This says to select all data in the To Do list table and to organize it by priority level, which is highest to lowest.

# Database Design

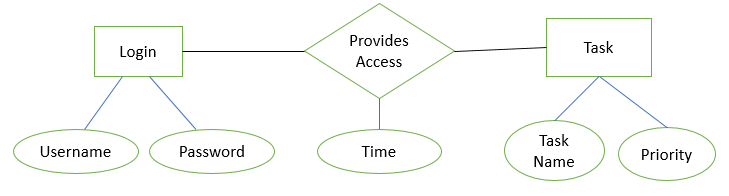


Figure 5: Entity-Relationship Diagram

The database diagram shown above depicts three tables that MySQL would hold for the application. The first table is Login which would require a username and a password. The second table would be labeled Task and it would have attributes such as task name and priority level. To connect the two tables to one another, another is created which is called Provides Access. This table would take the information from the Login and Task table. This table also has its own individual attribute called Time. When a user logs into the web application successfully, access will be provided. At that time, the time will be taken and inserted into the table. This will allow a better understanding of when a user logged in.

# Class Diagram

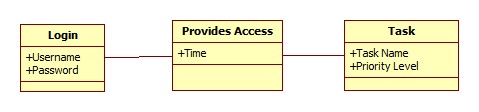


Figure 6: Class Diagram

The class diagram portrays the same information shown in the entity-relationship diagram but in a different format. In this website, there are three classes and each has its own attributes. The classes interact with one another as well.

# Test Case

There were three operations which needed to be tested for this application.

|  |  |
| --- | --- |
| Test Cases | Pass/Fail |
| View | Pass |
| Add | Pass |
| Delete | Pass |

Table 1: Test Cases

The first test case was to view the To Do list. To do this, clicking on the view button redirects the user to another webpage. This page has a table with all the data in the database table.

The second test case was to add a task to the table. To check if this function is working correctly, the user would enter data into the text field provided and click on the ‘Add’ button. To see if the entered data was inserted into the database, the user would refresh the view page. This should show the newly entered task name.

The final test case is to delete a task from the list. To test this, the user would enter the name of the task into the provided text field. After clicking the ‘Delete’ button, check the view webpage to see the data. This page should show a table with all tasks besides the one that was just deleted.

When performing all three test cases, the results came out to be successful. This ensured that the database was integrated into the website through phpMyAdmin.

# Timing

The total number of hours spent on this project about 21 while the estimated time was 14. This difference in time is mainly because of additional research on how to create and design the webpage, to understand the different terms and diagrams, and to solve issues relating to MySQL and XAMPP. A breakdown of each task that this project required is below.

Name of the task: Research

Estimated No of hours: 2

Actual time spent: 1

Notes: Spent this time researching about different forms of diagrams and understanding the difference between functional and non-functional requirements. Also, briefly researched about the steps needed to create a client/server architecture using XAMPP.

Issues: There were no issues.

Name of the task: Explore MySQL and create table

Estimated No of hours: 0.5

Actual time spent: 2

Notes: Explored MySQL and created simple schemas to reverse and forward engineer. Reviewed simple SQL commands and how to insert data into a table.

Issues: Initially had problems accessing MySQL workbench and spent an hour trying to see if it was because I had the wrong password or if MySQL was using a different port.

Name of the task: Create TODO website

Estimated No of hours: 5

Actual time spent: 10

Notes: Spent a total of 8 hours creating the To Do website. During those hours, I implemented the necessary functions using PHP, HTML, and SQL. I also explored different ways to format and design the website using Bootstrap.

Issues: I originally created the website and used EasyPHP to access the database. After reading the requirements over, I noticed that we were supposed to use a WAMP or XAMPP stack. I had XAMPP already installed but the options for MySQL were disabled. Researching and fixing this issue took about 2 hours. Once XAMPP started working, I took the files I originally created and copied it over to the XAMPP folder. Test cases were then used to check the integration of the website with the database. The status of each test case was successful.

Name of the task: Documentation

Estimated No of hours: 6

Actual time spent: 7

Notes: While writing the document, I spent time in between to research topics I was not familiar with and to understand how to create some of the diagrams. Researching, writing, and editing were all a part of this time portion.

Issues: There were no issues.

Name of the task: Upload to Github and create video presentation

Estimated No of hours: 0.5

Actual time spent: 1 hour

Notes: I first spent a few minutes to familiarize myself with Github and looking at short tutorials. After uploading my files to Github, I started working on the video presentation. Majority of the time was used looking for a way to record the screen.

Issues: There were no issues.