**Product Documentation**

**To-Do Application**

Version 1.0

**DOCUMENT OWNER**: Alaa Al-Naber

**Document History:**

|  |  |  |  |
| --- | --- | --- | --- |
| Timestamp | Version Number | Updated-By | Comments |
| 1/10/18  9:39 AM | 1.0 | Alaa Al-Naber | Initial Draft - Creating and setting up the documentation title page, document layout, table of contents, and content for each requirement/diagram/case |

**Table of Contents**

**1. INTRODUCTION**

* 1. Purpose of Document………………………………………………………………....1
  2. Application Overview………………………………………………………………...1
  3. Developed on Chrome……………………………………………………………......1

**2. REQUIREMENTS**

2.1 Functional requirements………………………………………………………………2  
2.2 Non-functional requirements……………………………………………………….....2

**3. DIAGRAMS**

3.1 System Architecture Diagram…………………………………………………………4   
3.2 Data Flow Diagram……………………………………………………………………5  
3.3 Sequence Diagram…………………………………………………………………….6  
3.4 Class Diagram…………………………………………………………………………9

**4. CASES**

4.1 Use Case……………………………………………………………………………...10  
4.2 Test Case

**5. DESIGN**

5.1 Database Design

**6. WORK-BREAKDOWN**

6.1 Details of work

**1. INTRODUCTION**

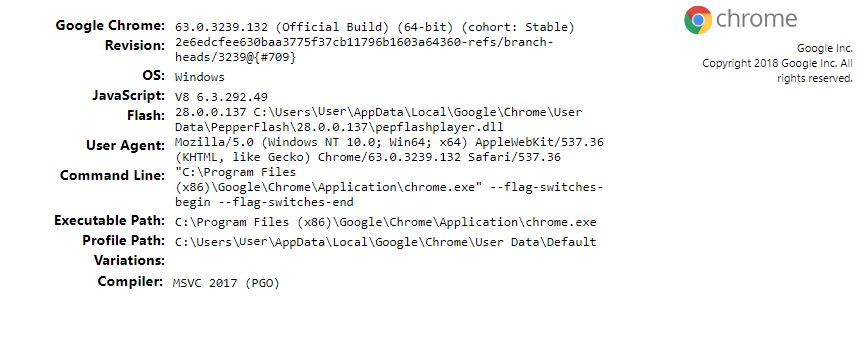
1.1 PURPOSE OF THIS DOCUMENT

This document’s purpose is to give background information of the todo-list application, and this includes requirements, diagrams, and cases. The documentation will define the functional and non-functional requirements. Diagrams included in this document are as follows; System Architecture, Data Flow, Sequence, and Class Diagrams. Use cases and Test cases will be given as well as a Database Design. This is meant as a guide for further development.

1.2 APPLICATION OVERVIEW

This application will allow you to create a task and mark its status. Totals are counted on the main page and this application is hosted on a XAMMP Stack. The todo-list app was developed on google chrome.

1.3 DEVELOPED ON CHROME



**2. REQUIREMENTS**

2.1 FUNCTIONAL REQUIREMENTS

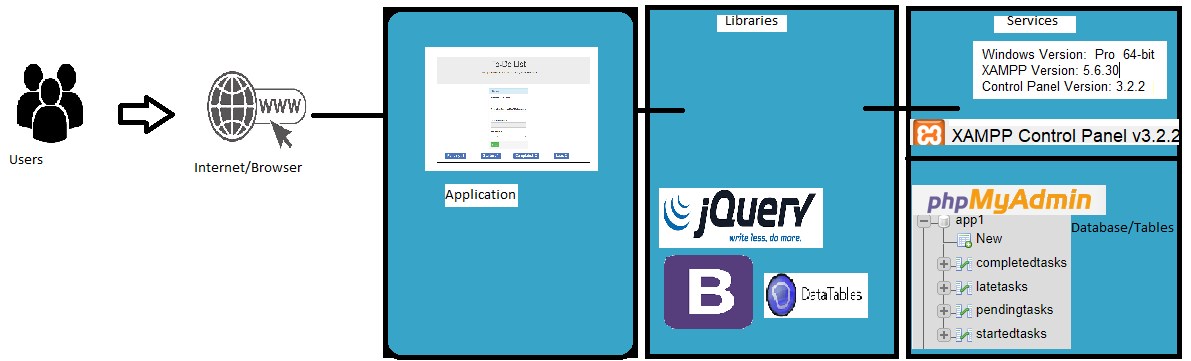
|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Requirement Statement | Must/Want | Comments |
| FR01 | The application must have a button to add tasks. | Must | Button can be on home screen with fields about the task. |
| FR02 | The application must have a button to delete tasks. | Must | Button on home screen. |
| FR03 | The application must have fields where a task can be entered. | Must | Fields are on home page. |
| FR04 | The application must allow someone to switch the status of a task. | Must | Use a select box to select from different status. |
| FR05 | A due date is required for each task. | Must |  |
| FR06 | Totals for each task status is counted. | Must | Add on the home page. |

2.2 NON-FUNCTIONAL REQUIREMENTS

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Requirement Statement | Must/Want | Comments |
| NF01 | This application should be run on a web browser that can support html5. | Want | HTML5 attributes such as maxlength are used. |
| NF02 | The operating system must be greater than windows XP. | Must |  |
| NF03 | The machine must be able to XAMMP stack. | Must | This is a must because there is no public domain for this web application and therefore cannot be hosted on a different machine. Unless you are using the application on the intranet. |
| NF04 | This is a freely distributable application. | Must |  |
| NF05 | The space of the website will be dependent on the user’s machine. | Must | Rows are stored in the database and rows contain information about the task. |
| NF06 | The website should be secure from injections. | Want | Since this app is not public this is not much of a must. |

**3. DIAGRAMS**

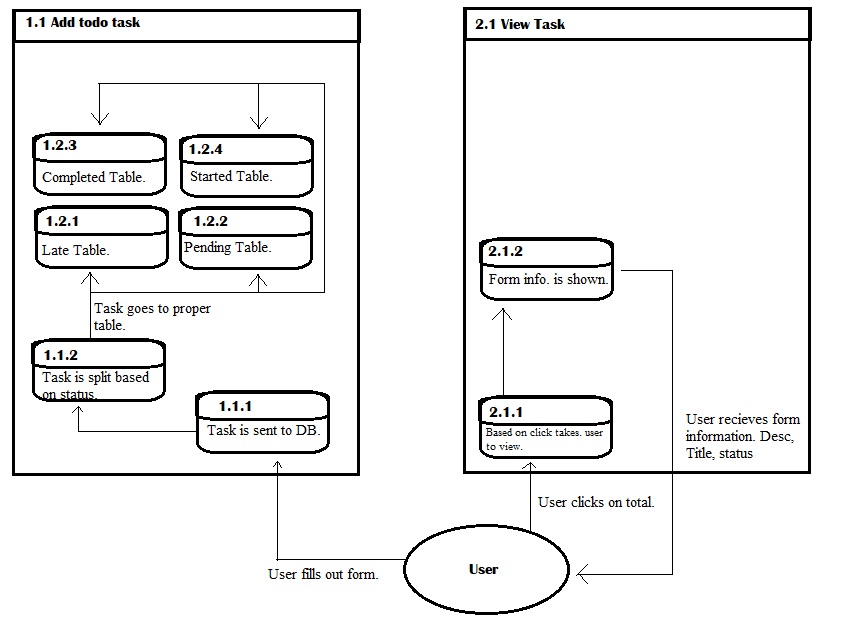
3.1 SYSTEM ARCHITECTURE DIAGRAM



The software requires a desktop with a minimum 512MB of ram, and hard drive space of 1024MB.

The software is built with bootstrap as the style library and scripts. JQuery as a scripting library, these scripts include Datepicker and DataTables. The front end was developed with HTML, and JavaScript is a scripting language used for actions. The backend that is used in this software is PHP 5.6.30. The server type is MariaDB and the version is 10.1.21.

3.2 DATA FLOW DIAGRAM



3.3 SEQUENCE DIAGRAMS

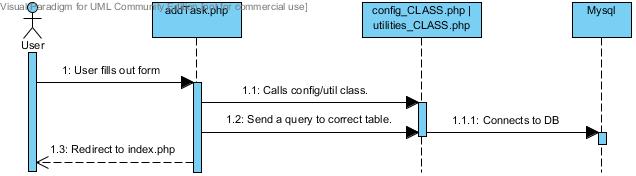


Figure 3.3.1 Adding a task.

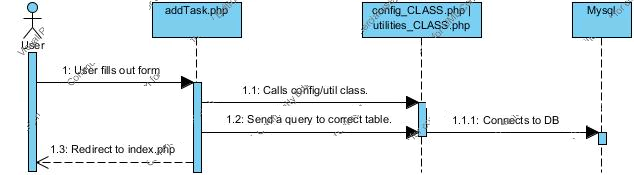


Figure 3.3.2 Viewing a task.

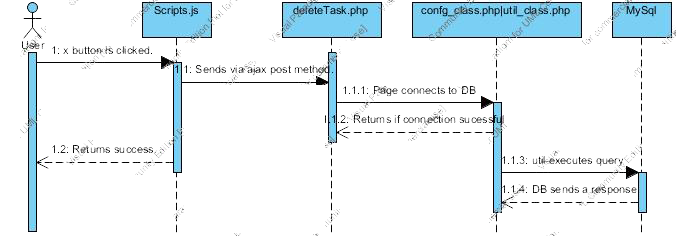


Figure 3.3.3 Deleting a task.

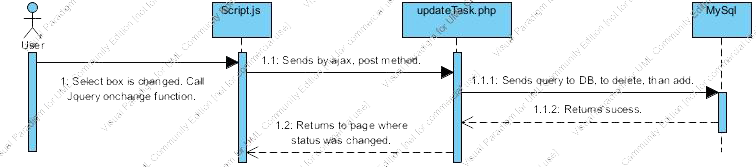


Figure 3.3.4 Updating a task.



Figure 3.3.5 Display the application Version Number.

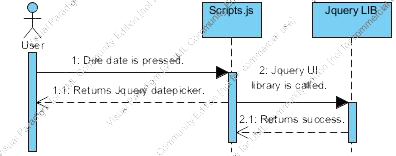
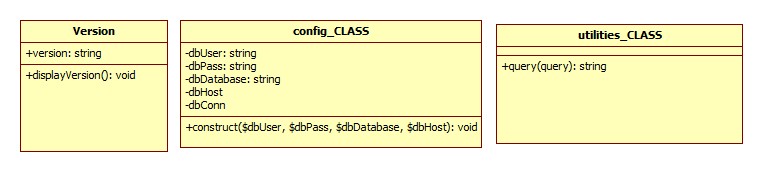


Figure 3.3.6 Datepicker

3.4 CLASS DIAGRAM



**4. CASES**

4.1 USE CASES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-1 | | | |
| **Use Case Name:** | User Adds a Task | | | |
| **Created By:** | Alaa Al-Naber | | **Last Updated By:** | Alaa Al-Naber |
| **Date Created:** | 1/13/18 | | **Last Revision**  **Date:** | 1/13/18 |
| **Actors:** | | User | | |
| **Description:** | | The user will fill the form with the following information: title, description, due date, and status. | | |
| **Trigger:** | | Users clicks on the submit button on the main page. | | |
| **Preconditions:** | | The task is not in the database. | | |
| **Postconditions:** | | Information is stored in the database. And the user is redirected to landing page. | | |
| **Normal Flow:** | | 1. User goes to the website. 2. The user fills the fields. 3. The user clicks on the submit button. 4. The information goes to the database. 5. User is redirected to the index page. | | |
| **Frequency of Use:** | | High – A user can input many tasks. | | |
| **Assumptions:** | | An internet connection is established and the user can access the main page. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-2 | | | |
| **Use Case Name:** | User Views a Pending Task | | | |
| **Created By:** | Alaa Al-Naber | | **Last Updated By:** | Alaa Al-Naber |
| **Date Created:** | 1/13/18 | | **Last Revision**  **Date:** | 1/13/18 |
| **Actors:** | | User | | |
| **Description:** | | The user will be allowed to see a pending task. | | |
| **Trigger:** | | Users clicks on the pending total. | | |
| **Preconditions:** | | No preconditions needed. | | |
| **Postconditions:** | | User is redirected to the pending page and there they can view the pending tasks. | | |
| **Normal Flow:** | | 1. User goes to the website. 2. The user clicks on the pending total. 3. The user is redirected to the pending page. 4. User sees all of their pending tasks. | | |
| **Frequency of Use:** | | High – A user will want to view their tasks often. | | |
| **Assumptions:** | | An internet connection is established and the user can access the main page. From the main page the user can view the pending task totals. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-3 | | | |
| **Use Case Name:** | User Views a Started Task | | | |
| **Created By:** | Alaa Al-Naber | | **Last Updated By:** | Alaa Al-Naber |
| **Date Created:** | 1/13/18 | | **Last Revision**  **Date:** | 1/13/18 |
| **Actors:** | | User | | |
| **Description:** | | The user will be allowed to see a started task by click on the started total on the index page. | | |
| **Trigger:** | | Users clicks on the started total. | | |
| **Preconditions:** | | No preconditions needed. | | |
| **Postconditions:** | | User is redirected to the pending page and there they can view the started tasks. | | |
| **Normal Flow:** | | 1. User goes to the website. 2. The user clicks on the started total. 3. The user is redirected to the started page. 4. User sees all of their started tasks. | | |
| **Frequency of Use:** | | High – A user will want to view their tasks often. | | |
| **Assumptions:** | | An internet connection is established and the user can access the main page. From the main page the user can view the started task totals. | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID:** | UC-4 | | | |
| **Use Case Name:** | Changing Status of a Task | | | |
| **Created By:** | Alaa Al-Naber | | **Last Updated By:** | Alaa Al-Naber |
| **Date Created:** | 1/13/18 | | **Last Revision**  **Date:** | 1/13/18 |
| **Actors:** | | User | | |
| **Description:** | | This feature allows a user to change the status of a task. | | |
| **Trigger:** | | User clicks on the select box in the Change status. | | |
| **Preconditions:** | | User is on the view page. | | |
| **Postconditions:** | | User is redirected back to the status that they were originally viewing. | | |
| **Normal Flow:** | | 1. User goes to view page of any status. 2. User goes to the change status column. 3. User clicks on the status select box of the row they would like to update. 4. The system removes the row from the original status and adds it to the new status table. | | |
| **Frequency of Use:** | | High – A user will want to change their tasks often. | | |
| **Assumptions:** | | An internet connection is established and the user can access the main page. From the main page the user can view the select box choices. | | |

**6. WORK-BREAKDOWN**

6.1 DETAILS OF WORK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name of Task | Est. Time Spent | Act. Time Spent | Notes | Issues |
| Front End Creation. | 3 hrs. | 2 hrs. | Most difficult issue was figuring out how to layout the application. | No coding issues, only creative issues. |
| Adding a task. | 1 hr. | 1 hr. | Needed to create a class to connect to DB. | No Major issues. |
| Editing the status of a task. | 3 hrs. | 4 hrs. |  | No Major issues. |
| Deleting a task. | 30mins. | 1hr | Added javascript to get the data-id. Sent it through ajax to php page. | Ran into a bug where first row will only be deleted while the rest do not get deleted. |
| Creating DB | 30mins | 10mins | Used phpmyadmin GUI to create DB. | No major issues. |
| Setting up Git/version control. | 20mins | 1hr. | Used github and SourceTree. | My first attempt at creating a git lead to me having to pull a file that doesn’t exist. I recreated the clone and that fixed the issue. |
| Counting the totals. | 1hr. | 3hrs. | Had to use mysql\_result built in php function in order to use the count function. |  |