|  |
| --- |
| Group 16 |
| The Project Maintenance Manual |
| Author: Robert Mouncer |

Config Ref: SE\_16\_MI\_01

Date: 26/01/2016

Version: 1.0

Status: Released

Department of Computer Science Aberystwyth University Aberystwyth Ceredigion SY23 3DB Copyright © Aberystwyth 2016

|  |  |
| --- | --- |
|  |  |

Contents

[1.0 Introduction 4](#_Toc441591921)

[1.1 Purpose of this Document 4](#_Toc441591922)

[1.2 Scope 4](#_Toc441591923)

[1.3 Objectives 4](#_Toc441591924)

[2.0 Program Description 4](#_Toc441591925)

[2.1 TaskerMAN 4](#_Toc441591926)

[2.2 TaskerCLI 4](#_Toc441591927)

[3.0 Program Structure 5](#_Toc441591928)

[3.1 TaskerMAN 5](#_Toc441591929)

[3.1.1 Folders needed 6](#_Toc441591930)

[3.1.2 addmember.php 6](#_Toc441591931)

[3.1.3 checkLogin.php 6](#_Toc441591932)

[3.1.4 connect.php 6](#_Toc441591933)

[3.1.5 createMember.php 6](#_Toc441591934)

[3.1.6 createTask.php 7](#_Toc441591935)

[3.1.7 deleteMember.php 7](#_Toc441591936)

[3.1.8 editTask.php 7](#_Toc441591937)

[3.1.9 home.php 7](#_Toc441591938)

[3.1.10 homeStyles.css 7](#_Toc441591939)

[3.1.11 index.php 7](#_Toc441591940)

[3.1.12 logout.php 7](#_Toc441591941)

[3.1.13 members.php 7](#_Toc441591942)

[3.1.14 membersEdit.php 8](#_Toc441591943)

[3.1.15 membersStyles.css 8](#_Toc441591944)

[3.1.16 menu.php 8](#_Toc441591945)

[3.1.17 processTask.php 8](#_Toc441591946)

[3.1.18 removeMember.php 8](#_Toc441591947)

[3.1.19 styles.css 8](#_Toc441591948)

[3.1.20 tasksStyles.css 8](#_Toc441591949)

[3.1.21 updateMemberInfo.php 8](#_Toc441591950)

[3.1.22 updateTasks.php 9](#_Toc441591951)

[3.1.23 viewTasks.php 9](#_Toc441591952)

[3.1.24 viewTask.php 9](#_Toc441591953)

[3.2 TaskerCLI 10](#_Toc441591954)

[3.2.1 MainFrame.java 11](#_Toc441591955)

[3.2.2 Load.java 11](#_Toc441591956)

[3.2.3 Task.java 11](#_Toc441591957)

[3.2.4 DatabaseConnect.java 11](#_Toc441591958)

[3.2.5 TaskerLogin.java 11](#_Toc441591959)

[3.2.6 TaskerPage.java 12](#_Toc441591960)

[3.2.7 TaskerEditor.java 12](#_Toc441591961)

[Change History 12](#_Toc441591962)

# Introduction

## 1.1 Purpose of this Document

This document is set out to answer any likely questions that programmers may have when maintaining the program.

## 1.2 Scope

This document describes the program created in detail so that maintainers and installers know which part of the program is likely to provide the answers to questions that they may have. The maintainers should be able to easily navigate this document to find the answers. It will explain the main functions and methods with pseudo code so that it is easy to understand.

## 1.3 Objectives

The objective of this document is to explain the programs that have been created for our group project. This document should include descriptions of all algorithms, files and explain in details the program structure. Improvements to both programs will be suggested for future use, but include hazardous areas that should be taken with precaution when changing.

# Program Description

## 2.1 TaskerMAN

TaskerMAN is a series of webpages that is used for the creation, allocation and monitoring of tasks. It uses a MySQL database to store and retrieve the relevant data. New users and managers can be added to the database. Only the managers can use the website to allocate the tasks.

## 2.2 TaskerCLI

TaskerCLI is a desktop application written in Java for team members to view their allocated tasks, the team members can only view their own task. The team members can report progress of their tasks and mark them as complete once they are done.

# 3.0 Program Structure

## 3.1 TaskerMAN Files



### **3.1.1 Folders needed**

**Folder Name**: \TaskerMAN\

**Purpose:** To store all relevant files of TaskerMAN in.

**Folder Name**: \TaskerMAN\Images\

**Purpose:** To store all images.

**Folder Name**: \TaskerMAN\Images\Profile Pics

**Purpose:** To store all images used for profile pictures.

**Folder Name**: \TaskerMAN\scripts

**Purpose:** To store all scripts needed for the website to run.

## 3.2 TaskerCLI Files

This is what the file store looks like:



### **3.2.1 DatabaseConnect.java**

**File Name:** DatabaseConnect.java

**Purpose:** This class is used to connect to our database.

**Dependency:** Task.java

**How it works:** The Java application synchronises with the database every five minutes, or if changes have been made to a task. If there is a connection established, it will check if any new tasks have been added or deleted on the server, then update local storage accordingly. It will then upload any changes that are done locally, and then update the matching tasks on the server with that information.

This is how the DatabaseConnect.java class is used to connect to the database at the start of use:



### **3.2.2 Load.java**

**File Name**: Load.java

**Purpose:** To save and load to and from the local storage.

**Dependency:** DatabaseConnect.java

**How it works:** When the **TaskerLogin.java** starts, it loads all the tasks currently stored in local storage into the java application and are stored as Task Objects in an arrayList. This is used to save the tasks from the database into local storage, on the user's computer. Tasks are saved before editing, after the task has been edited and every five minutes, to ensure that the tasker system is always updated.

This is how the save method works in the load.java file:



This is how the load method works in the load.java file:



### **3.2.3 MainFrame.java**

**File Name**: MainFrame.java

**Purpose:** This class is the main controller of the program.

**How it works:** It starts and initialises the program by calling **TaskerLogin.java.**

### **3.2.4 Task.java**

**File Name:** Task.java

**Purpose:** Creates an object that is defined by the task properties.

**How it works:** It is an object class that holds all the properties for tasks, when a new task needs to be created this class is used.

### **3.2.5 JListListener.java**

**File Name:** JListListener.java

**Purpose:** Whenever the JList is called the list should be updated.

**How it works:**

### **3.2.6 TaskerEditor.java**

**File Name:** TaskerEditor.java

**Purpose:** Creates the editor window for tasks

**Dependency:** TaskPage.java

**How it works:** .It creates a large edit panel, which is populated with the full task description and comments and allows the user to edit whatever is in there. Once finished the user must then press the submit button, which saves it to local storage.

### **3.2.7 TaskPage.java**

**File Name:** TaskPage.java

**Purpose:** This class displays all the information of all tasks. The information is displayed in a table and when a task is selected its full description will be displayed next to it in a panel. The user also has the ability to search the table, to help find a specific task. Once a task has been selected the user can press a button below the full task description to edit the task.

**Dependency:** TaskerLogin.java

**How it works:**

### **3.2.8 TaskerLogin.java**

**File Name:** TaskerLogin.java

**Purpose:** This class creates the graphical user interface for the login part of the application.

**Dependency:** MainFrame.java

**How it works:** It requires a username and password, which is compared with values in the database. If authentication is successful the user is directed to **TaskerPage.java.** However if the user is unsuccessful then they will be given the option to use offline mode which uses the local storage while checking for database connection.

### **3.2.9 TaskerLogin.java**

**File Name:** ChooseLocation.java

**Purpose:**

**How it works:**

### **3.2.10 chooseLocation.java**

**File Name:** ChooseLocation.java

**Purpose:**

**How it works:**

# 4.0 Algorithms

# 5.0 The Main Data Areas

# 6.0 Files

# 7.0 Interfaces

# 8.0 Suggestions for improvements

Add the feature to allocate a task to multiple people, once one person has marked it as complete it will notify the other members allocated. This would be useful if you needed personal documents from each of the people.

# 9.0 Things to watch for when making changes

# 10.0 Physical Limitations of the program

One physical limitation is the time it takes to upload and download the tasks. We tested uploading over 4000 tasks and that could only upload as fast as the university network and the database could go, it took even longer to download into the java application. We think that this would be an extremely rare case that one person would have anywhere near 500 tasks to complete. When only loading 50 tasks it was almost instant, you wouldn’t notice

While testing over 4000 tasks on the java application, TaskerCLI managed to use 130MB of RAM. Again it would be a very rare case that one person would have this many tasks, but when we were testing with 50 tasks it also used 130MB of RAM, it seems to be a constant 130MB, but if a much larger amount of tasks were created then more memory would be used, again this is a very rare case.

Another limitiation is the storage for the file created from TaskerCLI. When testing with 4000 tasks it created a file that was 110MB but this will obviously vary depending on the task title and task description. But when testing with 50 tasks the memory used was only 3kb. This is a more reasonable amount of memory. People won’t be allocated up to 4000 tasks and won’t need this memory. This is why a user would only need a recommended maximum of 2MB for storage.

The website, TaskerMAN, needs storage on a hosting server, it is not much memory as it is less than 2MB. So as long as this is available the website will run without problem. Though profile pictures will take up additional space depending on the size.

The MySQL database also has a limit to how much data can be stored on it. This is usually 2GB which can store a lot of data and should never get anywhere near this limit. If this is reached then another database would need to be set up, depending on if its tasks or members.

Another limitation would be if the server or hosting page was down, users would not have any access to the database and could not download newly allocated tasks. They would only be able to work in offline mode.

# 11.0 Rebuilding and testing

# Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF  No. | Date | Changes Made To Document | Changed By |
| 1.1 | N/A | 2016-01-26 | Ready for review. | Robert Mouncer – rdm10 |
| 1.2 | N/A | 2016-01-26 | Ready for release. | Robert Mouncer –rdm10 |
| 1.3 | N/A | 2016-01-28 | Ready for Review | Robert Mouncer –rdm10 |