Software Engineering Group Project
Maintenance Manual

Author:	Ben Dudley, David Fairbrother, Jonathan Englund, Josh Doyle, Liam Fitzgerald, Maurice Corriette, Oliver Earl, Tim Anderson
Config Ref:	SE_05_MAINT_01
Date:	2016-02-14
Version:	1.0
Status:	Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document contains essential information for the maintenance of TaskerCLI, TaskerMAN and TaskerSRV [1].

1.2 Scope

This information in this document pertains to future development, maintenance, installation and testing of the Tasker system.

1.3 Objectives

The reader of this document should be able to identify the offending code for future bugs, or the best location to add new features. It also forewarns of any potential pitfalls for the unwary programmer and how to test functionality of updated components.

2. TASKERCLI MAINTENANCE MANUAL

2.1 Program Description

TaskerCLI is a desktop application which allows users receive tasks. It allows users to change the task status, have multiple elements per task and add or edit elements for that task. It works both offline and online and will automatically synchronise to the database.

2.2 Program Structure

TaskerCLI has two groups of classes, GUI classes such as MainWindow and LoginWindow and logic classes such as Database and Task.
GUI classes handle program start up, display and shutdown. Logic classes connect to the database and provide storage of data such as members or tasks.

2.3 Algorithms Used

TaskerCLI populates the main display from “taskSaveFile.txt” (described below). This allows it to display data both online and offline. To achieve this it parses the save file into the respective objects used within the program. In the event we cannot parse the file we discard the contents and ask the database to retrieve a new copy.

The database automatically synchronizes and attempts to reconnect using Timers, when fired these trigger methods to contact the database. When data is successfully retrieved from the database it is converted into a MemberList or TaskList. The TaskList is passed to MainWindow which handles saving and updating the display whilst MemberList is saved by the database class to a flat text file.

When a task is edited to ensure it does not discard the user's changes which are in progress automatic synchronization is paused. When the user closes the editing window it immediately resumes after saving their changes.

2.4 Main Data Areas

All Member Objects are held within a MemberList Object, and all Task Objects are held within a TaskList Object. These list Objects are essentially just ArrayLists that have been abstracted into their own class. The vital methods used for an ArrayList, such as get and add, are still accessible through the methods associated with the MemberList and TaskList Classes.

Two enumerations are used in TaskerCLI. These two enumerations are used to represent the state of the connection between TaskerCLI and the TaskerSRV server, and the completion status of a single Task.

The database object holds a JDBC connection to the database which is open during the operation of the program. The details used to establish the connection are also stored within the database object to allow automatic reconnection. It also holds a reference to the main window which allows it to display error and warning messages.

2.5 Files

- **Save Files**
    - TaskerCLI uses four save files to save member and tasks data, TaskerSRV credentials to enable automatic logins and tasks pending synchronisation to the TaskerSRV database. These files need to be located in the same directory as the TaskerCLI .jar file.
    - **Member Save File**
        - memberSaveFile.txt
        - Holds the names and email addresses of all the Members that are to be granted access to TaskerCLI
        - If the file does not exist, it will be created on start-up and automatically populated when TaskerCLI is connected to TaskerSRV
        - If the contents of the files becomes corrupted, its contents will be overwritten with Member data downloaded from TaskerSRV
    - **Task Save File**
        - taskSaveFile.txt
        - Holds the data associated with all tasks stored in TaskerSRV
        - If the file does not exist, it will be created on start-up and automatically populated when TaskerCLI is connected to TaskerSRV
        - If the contents of the file becomes corrupted, its contents will be overwritten with Task data downloaded from TaskerSRV
    - **TaskerSRV Credentials Save File**
        - connSaveFile.txt
        - Holds the credentials of the TaskerSRV database
        - Credentials are saved when the user connects to the TaskerSRV database
        - If the user changes the credentials from within TaskerCLI, the save file will be updated automatically
    - **Pending Tasks File**
        - pendingSaveFile.txt
        - Saves Tasks that could not be synced with TaskerSRV so they can be synced when TaskerCLI is next connected to TaskerSRV
        - Pending Tasks are added to the file automatically and are removed when they are synchronised with TaskerSRV

2.6 Interfaces

The program uses a MYSQL JDBC driver to communicate with the database. As different JDBC drivers use different connection strings a MYSQL database must be used with this program. It can use a custom port for the MYSQL database but if one is not specified it uses the default port of 3306.

2.7 Future Improvement

The program’s usability can be further improved by displaying clearer and concise error messages describing what went wrong and how the user could fix it. Currently the program throws a dialog box if an automatic reconnect fails, as the user cannot change this behaviour it should not throw a dialog box.
Additionally a colour indicator could also show the current state of the connection and many text entry boxes don’t accept enter requiring the user to use their mouse.

2.8 Potential Pitfalls

Extra care must be taken editing methods which save, load, get and set a TaskList or MemberList. If the save format is not adjusted correctly the program will not display any tasks or allow login as it will keep discarding the file. All get and set methods are threaded, this can lead to unforeseen race conditions. Care needs to be taken if the order of these methods need to be serialised as they execute in their own thread and don’t return to the caller on completion.

Some of the tables used to display data in TaskerCLI appear to allow editing of cells. This, however, is not the correct method of editing data and any data entered into these cells will not be saved locally or synchronise with the TaskerSRV database. When editing Task Comments, selecting a Task Element from the table will display the Task Element in the two text boxes above the table. It is here that the Task Element Comment should be changed. Clicking the Submit button will then temporarily submit the changes, and clicking the Save button will commit changes locally.

2.9 Limitations

TaskerCLI requires very little processing power it has been tested and used with a 200 MHz processor
The program recommends 128MB of RAM with 64MB being minimum. Below this amount excessive swapping will occur severely degrading performance.
10MB of disk space is recommended whilst 4MB is minimum for the application and associated storage files.
The target system must also have a Java Runtime Environment installed for the program to function.

2.10 Rebuilding and Testing

To rebuild the project two external libraries are required: Swing MIGLayout 4.0 and MYSQL JDBC driver. Once added to the classpath the program should build. The program functionality is tested using the testing specification [2]. This is due to bugs mostly arising from race conditions which JUnit cannot easily detect.

3. TASKERMAN MAINTENCE MANUAL

3.1 Program Description

TaskerMAN is an online application (a website) that gives managers complete control over the users and tasks stored on the TaskerSRV database. It allows the creation, modification and deletion of users and tasks, including each task's individual elements.

Unlike TaskerCLI, TaskerMAN is only accessible by managers and also requires an Internet connection.

3.2 Program Structure

TaskerMAN is broken up into three sections - interfaces, logic and supporting files. The interfaces consist of three main files, 'index.php', 'taskerman.php' and 'users.php' - with the exception of 'index.php' which is the program's login page, these files use PHP includes to include other PHP files containing UI elements, such as modal windows for adding users, etc.

Pages such as 'editTask.php' which provides the code functionality for editing a task or 'connect.php' which provides a persistent database connection are logic files. They contain little to no HTML and are usually the POST target for forms contained on modal windows.

Supporting files include form validation JavaScript, stylesheets and images that are used by the program.

TaskerMAN is written procedurally, and therefore does not have classes, but does have global functions contained within the 'init.php' file, which is referenced by all files in the application.

3.3 Algorithms Used

There are no advanced algorithms used in TaskerMAN - however one of the most interesting or noteworthy is that which generates input textboxes with unique IDs based on a user defined number of elements.

- Take the number the elements the user wishes to add
- Use that number to construct a for loop
- Print input boxes with IDs based on the current loop iteration

3.4 Main Data Areas

The main data storage area used by the program are the files generated by the PHP Session system. These are server-side files stored in the tmp directory. While this is potentially unsafe - this could easily be reconfigured to any directory, including a directory outside of publicly accessible web folders. It is simply configured this way to ensure compatibility with Information Services servers.

3.5 Files

3.5.1 Interfaces:

- footer.php
  - Contains the footer for the TaskerMAN interface
- header.php
  - Contains the header for the TaskerMAN interface, including the navigation.
- index.php
  - Login page - the first page that users land on when accessing TaskerMAN
- meta.php

    - Contains HTML metadata and any non-PHP includes that are accessed sitewide, such as the CSS stylesheet.
- taskAdd.php
    - Modal window - provides functionality so that the user can add tasks to TaskerSRV
- taskEdit.php
    - Modal window - provides functionality so that the user can edit an existing task specified in the TaskerMAN main interface
- taskerman.php
    - The main TaskerMAN file - the Task view of the program. The bulk of program functionality is here - it uses PHP includes to load modal windows and other interface elements.
- taskView.php
    - Modal window - displays in detail the selected task
- userAdd.php
    - Modal window - provides functionality so that the user can add users to TaskerSTV
- userEdit.php
    - Modal window - provides functionality so that the user can edit an existing user specified in the TaskerMAN user interface
- users.php
    - The Users view of the program
- addElements.php
    - Modal window - allows the user to add a predetermined amount of task elements and their comments to a task
- additionalElement.php
    - Modal window - allows the user to add an additional element to an existing task

3.5.2 Logic

- init.php
    - One of the most important files in TaskerMAN. This file provides access to global functions, such as those responsible for error handling, provides a site-wide database connection and is responsible for providing application persistence by configuring a PHP session. This file is called by almost every PHP file.
- taskAddGateway.php
    - Acts as a gateway between the Add Task and Add Element modal windows and provides important data sanitisation
- taskDelete.php
    - Deletes the selected task from TaskerSRV
- userDelete.php
    - Deletes the selected user from TaskerSRV
- addEntry.php
    - Adds a task and its respective task elements/comments to TaskerSRV
- addExtraElement.php
    - Adds an extra element to an existing task
- addUser.php
    - Adds a user to TaskerSRV
- config.php
    - Site-wide configuration file, read by init.php
- connect.php
    - Configures and establishes a database connection
- editTask.php
    - Modifies an existing task entry in TaskerSRV
- editUser.php
    - Modifies an existing user in TaskerSRV
- elementDelete.php
    - Deletes all elements from an existing task

3.5.3 Supporting Files

- style.css
    - Stylesheet
- entryValidation.js
    - Validation for the login / index.php
- validation.js
    - Validation procedures for all remaining forms

3.6 Interfaces

TaskerMAN uses the PDO_MYSQL driver (that which itself implements the PDO interface) to communicate with TaskerSRV. Custom ports can be specified.

3.7 Future Improvement

One of the key requirements was accidentally left out during initial development: the ability to filter data displayed in the TaskerMAN main interface and sort it based on column values, namely sorting tasks based on their status. While this was a genuine small oversight, realistically this would be easy and low-risk to implement into the next version of the software, and could be done in a realistic timeframe of less than a few days.

TaskerMAN would however benefit highly from the refactoring of code in the future as there are some significant smells present in the source code, namely code repetition. Designating functionality to specific functions and reducing the amount of files would make software maintenance easier for future development. It would also be feasible to refactor the code to fit the object-oriented programming paradigm that PHP fully supports.

The use of AJAX was discussed as a means of dynamically updating both the TaskerMAN interface and passing data between modal windows without the need to POST data manually to the server. While this was deemed unnecessary and would add an extra layer of complexity to the software, experienced developers could take this into consideration.

One final major improvement identified by the developers is the lack of proper responsive design. While TaskerMAN behaves as expected in all major browsers, including mobile browsers, (with some presentational hiccups present when running under Internet Explorer), it was not designed to adapt its presentation based on certain characteristics, such as the low resolutions naturally present on mobile devices. A redesigned stylesheet and HTML layout would look to incorporate these features for maximum compatibility and ease of use.

3.8 Potential Pitfalls

As the layout of the program is not always necessarily clear, those maintaining the software must be careful to ensure that they do not break any dependencies as these will cause problems that cascade throughout the software. Regression testing will be important as code is refactored.

3.9 Limitations

As the majority of TaskerMAN is server-side, the main limitations imposed include an active Internet connection, a managerial account and a relatively modern web browser. HTML5 support is a plus as it affects the presentation of modal windows and the use of HTML5 validation, but is not a requirement.

As TaskerMAN is web based, it does not behave differently based on the user's operating system. Mobile devices and low resolutions are for the most part supported, but a minimum of 800x600 is generally recommended.

While the JavaScript and HTML5 was vigorously tested to ensure no invalid input could be sent to the program and that there is input sanitisation in use throughout the program to ensure no malicious input can make it to

TaskerSRV - it is by no means bulletproof. An experienced hacker or penetration tester would likely be able to bypass it easily, and of course, the system is more vulnerable if JavaScript is disabled, or if HTML5 is not supported by the user's browser.

3.10 Rebuilding and Testing

To deploy a new installation the requirements specified in the deployment description need to be met on the target server. Copy the tar containing TaskerMAN and automated install script and execute the script. This will setup the files, permissions and create the directory structure needed.

4. TASKERSRV MAINTENANCE MANUAL

4.1 Program Description

A MySQL database which holds tasks, users and their associated details. Both TaskerMAN and TaskerCLI synchronise to this database.

4.2 Tables and schema

TaskerSRV uses three tables to store data they are:

- tbl_users
  - Email (Primary Key, VARCHAR(45))
  - FirstName (Not NULL, VARCHAR(15))
  - LastName (Not NULL, VARCHAR(15))
  - IsManager (Not NULL, INT(11))
- tbl_tasks
  - TaskID (Primary Key, Not NULL, Auto Increment, Int(11))
  - TaskName (Not NULL, VARCHAR(45))
  - StartDate (Not NULL, DATE)
  - EndDate (Not NULL, DATE)
  - Status (Not NULL, INT(11))
  - TaskOwner (Foreign Key tbl_users->Email, Default NULL)
- tbl_elements
  - Index (Primary Key, Not NULL, Auto Increment, INT(11))
  - TaskDesc (Not NULL, VARCHAR(45))
  - TaskComments (VARCHAR(45), Default NULL)
  - TaskID (Foreign Key tbl_tasks->TaskID, Default NULL)

4.3 Rebuilding and Testing

In the event the database needs rebuilding the databaseSetup.sh script can be used. This can be used automated with command line flags or interactively. If a database already exists it will ask for confirmation before performing the operation and will reset the database to the original state. After resetting the login the only user accessible (unless test data is entered) will be “manager@example.com".
The script additionally will provide test users and test data if instructed to which can be used to test other applications such as TaskerMAN and TaskerCLI.

REFERENCES

[1] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.11 1.9 - Producing a Final Report,* Aberystwyth University: Software Engineering Group Project, 2016.

[2] M. C. O. E. David Fairbrother, *Testing Specification 2.1,* Aberystwyth University: Software Engineering Group Project, 2016.

[3] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	2016-02-14	Original Version	DAF5