Group 16

The Project Maintenance Manual

Author: Robert Mouncer, Greg Sharpe, Richard Price-Jones

Config Ref: SE_16_MI_01

Date: 26/01/2016

Version: 1.2

Status: Release

Contents

1.0 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.

2.0 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

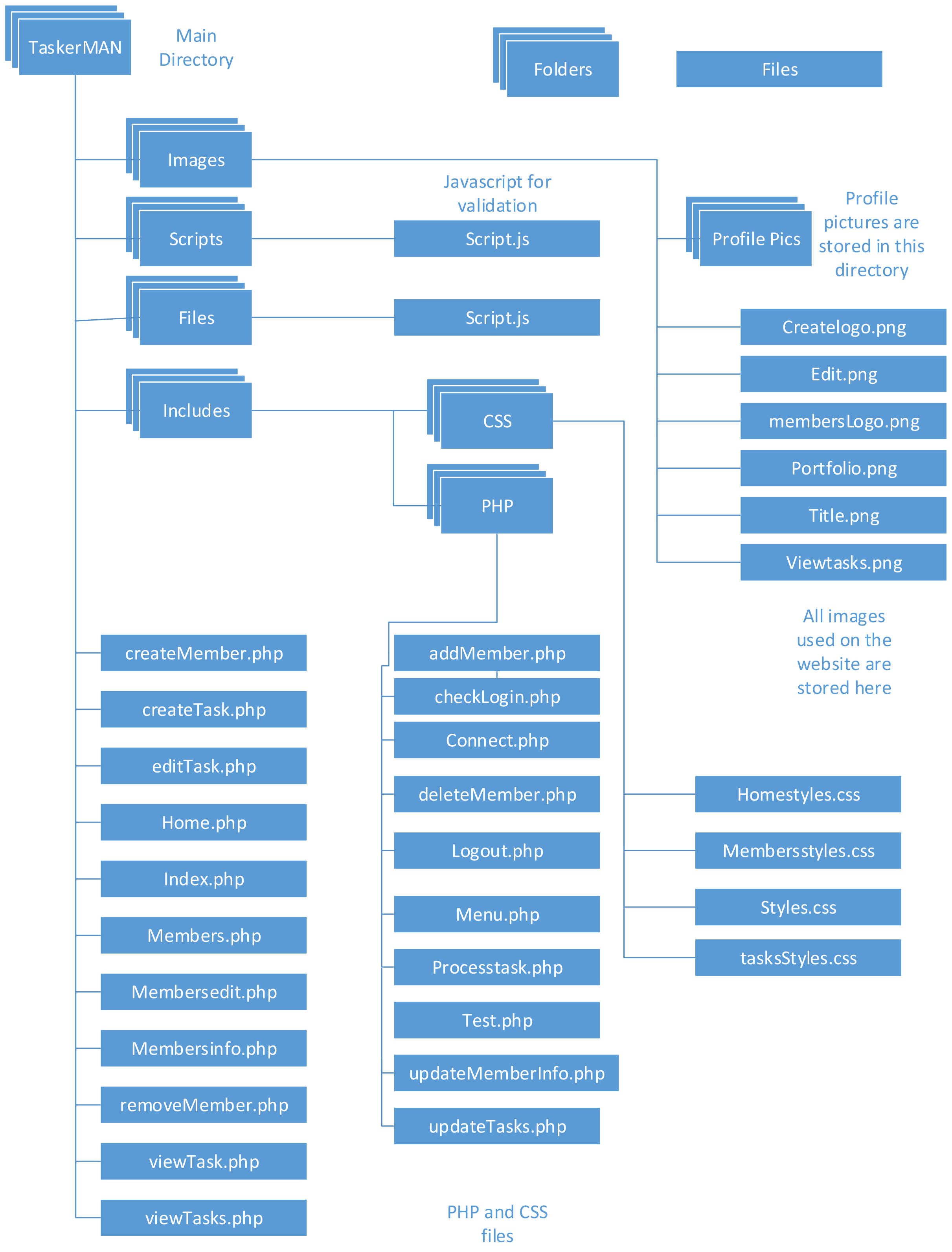
TaskerCLI is a desktop application written in Java for team members to view their allocated tasks, the team members can only view their own task. The team members can report progress of their tasks and mark them as complete once they are done.

2.3 TakerSRV

TaskerSRV is the MySQL database that holds all the information that the two programs need to operate correctly.

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.

3.1.2 Home.php

File Name: home.php

Purpose: The purpose of the page is to give the user a navigation between the pages. It allows the user to choose which page they would like to process onto. The options are 'Members', 'Create Task' and 'View Tasks'.

Functions needed: There is no php functions used on this page.

URL that uses the file: taskerMAN/home.php

3.1.3 Menu.php

File Name: menu.php

Purpose: The purpose for the menu.php file is to have the navigation bar in one place, to enable us to still make changes to it without applying all them changes throughout every page which makes it more time consuming

Functions Needed: The functions we will be using on this page are session_start() to run along side another function called isset, these together will check if the user is logged in. If they are not, it re-directs them to the login page.

URL that will use the file: this file will be called on every web page file that is not the login page.

3.1.4 Members.php

File Name: members.php

Purpose: The primary purpose is to display all of the users on the database and create links to each member's further information. There are also two buttons on this page an 'Add' and 'Remove' they will be used to add or remove members.

Functions Needed: We will use a while function within this file to populate a table from the 'members' table in the database.

URL that will use the file: taskerMAN/members.php

3.1.5 MembersInfo.php

File Name: membersInfo.php

Purpose: The purpose of this page is to give the user further information of a member, this also gives the user an option to edit the member.

Functions needed: There will be no php functions used on this page.

URL that will use the file: taskerMAN/membersInfo.php

3.1.6 MembersEdit.php

File Name: membersEdit.php

Purpose: The purpose of this page is to provide the user with opportunity to edit the name and e-mail, set the member an Admin or set the member with a display picture.

Functions needed: There will be no php functions used on this page.

URL that will use the file: taskerMAN/membersEdit.php

3.1.7 UpdateMemberInfo.php

File Name: updateMemberInfo.php

Purpose: The purpose of this file is to commit the changes made by the user from the 'membersEdit.php' file.

Functions Needed: We will need a function to filter the different variables, such as email, password and name. We will use filter_var function for this. We will also need to encrypt the password, for this I will use the 'hash' function to encrypt the passwords.

URL that will use the file: taskerMAN/membersEdit.php (once submitted)

Pseudo:

```
Define memberName from POST value

Define memberEmail from POST value

Define memberID from POST value
Define password from POST value


Update database members table with POST values


If Picture data is not empty Then
    Define newname as Picture new location
    If File already exists Then
        Delete Existing File
    End If
    Move uploaded File
End If


Redirect to “membersInfo.php”
```

3.1.8 CreateTask.php

File Name: createTask.php

Purpose: This page’s purpose is to allow a user to create a task for a member, adding in all the info and then having submit buttons.

Functions Needed: There will only be one php function in this file, it will be a while loop and it will be used to fill a dropdown box will the member’s names.

URL that will use the file: taskerMAN/createTask.php

3.1.9 ProcessTask.php

File Name: processTask.php

Purpose: The purpose of this page is to commit the new task and insert it into the database.

Functions Needed: We will need to again, filter the title and comments inputs of the new task. We will use filter_var for this. We will also have to run an update statement which will ‘$_POST’ the information from the previous file. The last function we will need is header, this will be placed last on the php file and will take the user to the viewTasks.php file.

URL that will use the file: taskerMAN/createTask.php

3.1.10 ViewTasks.php

File Name: viewTasks.php

Purpose: The purpose of this page will be to display all of the tasks which have been set by a member at one point. The tasks will be displayed in a white box with two options on the right side. These options will be 'View' and 'Edit'.

Functions Needed: We will need only one function for this page, that will be a loop to select and display all of the Tasks within the database.

URL that will use the file: taskerMAN/viewTasks.php

3.1.11 ViewTask.php

File Name: viewTask.php

Purpose: The purpose of this page is to give the user/member more information on the task they have clicked on. It will also display comments.

Functions Needed: There will only be one function for this page that will be to run a while loop to select and load the information based upon the task clicked (task id isset function may need to be used).

URL that will use the file: taskerMAN/viewTask.php

3.1.12 EditTask.php

File Name: editTask.php

Purpose: The purpose of this page is to allow the member/user to edit the task that they have chosen.

Functions Needed: We will use two different functions, the first being is isset this checks the taskID of the task you clicked on, on the previous page then loads all other data in. The second being a loop that runs to select members from the 'members' table and places them into a dropdown box.

URL that will use the file: taskerMAN/editTask.php

3.1.13 UpdateTasks.php

File Name: updateTasks.php

Purpose: The purpose of this file is to update the tasks table within our database, it is done by using an SQL update statement and posting the data from the previous page.

Functions Needed: The functions we will need to complete this is filter_var to filter and rid of special characters in the comment and title fields as there are no need for the characters in this case.

URL that will use the file: taskerMAN/editTask.php (on submission)

3.1.14 CreateMember.php

File Name: createMember.php

Purpose: The purpose of this web page is to allow the current user to add another member to the database. This will be completed by selecting a button on the members.php page. As a user you will be able to add the Name, Email, Password and a Profile picture of the new menu.

Functions Needed: for this page, we will need a POST method form to post the information about the new member to the following page. We also will need to have checks in place to not have duplicates in the database. We will use isset to check if the data that the user/member is trying to enter is already on there.

URL that will use the file: taskerMAN/createMember.php

3.1.15 addMember.php

File Name: addMember.php

Purpose: The purpose of this file will be to process the createMember.php page, meaning that all the information that the user has entered into the previous page inputted into the database table.

Function Needed: We will need to use a filter_var function, to filter the email, password and name to rid of the special characters. We will also need to encrypt the password to keep it secure, we will do this by using the hash function. Another function we will need is, a insert MySQL statement this will insert all of the new data into the database, by being posted to this page from previous pages.

URL that will use the file: taskerMAN/createMember.php (on click)

Pseudo:

```
Define memberName from POST value
Define memberEmail from POST value
Define password from POST value

Insert into members table in Database POST values

If Picture data is not empty Then
    Define newname as Picture new location
    If File already exists Then
        Delete Existing File
    End If
    Move uploaded File to newname location
Else
```

```
    Copy default Picture to newname location
End If

Redirect to “members.php”
```

3.1.16 CheckLogin.php

File Name: checkLogin.php

Purpose: The purpose of this page is after the user has logged in, another check will be made to ensure no problems.

Functions Needed: We will need to use filter_var to filter the username and password, we will need to encrypt the password.

URL that will use the file: taskerMAN/index.php

Pseudo:

```
Define username from POST value
Define password from POST value

If username is empty OR password is empty Then
    die()
End If

Query Database with select all from members Table where username and password match

If number of rows is greater than zero Then
    Start the session
    Redirect to “index.php”
Else
    Print “Login Failed”
End If
```

3.1.17 Connect.php

File Name: connect.php

Purpose: The purpose of this file is to create a link between the MySQL database held on the university server and our website, by storing the credentials of the database table in another file it increases security and allows us to cut down on the amount of code.

Functions Needed: There will only be a header function within this file,this will locate to the error.php file if an error occurs. All other functions within this file are MySQL functions. The first makes the connection using the provided database password and database user. The other is used to check if we can reach the database. If not the error.php file is called.

URL that will use the file: Every page will use the connect.php file

3.1.18 Error.php

File Name: error.php

Purpose: The main use of this page is to display an error message when the database table cannot be loaded. It will show up every time connection is lost between you and the database.

Functions Needed: The are no php functions for this file.

URL that will use this file: File is used on most pages when an error occurs.

3.1.19 Logout.php

File Name: logout.php

Purpose: The only main purpose of this file is to logout the user, it will do this by destroying the current session which involves your username and password.

Functions Needed: We will need a session start function, to start the session. A header function to relocate to the new file once complete, and a session destroy to remove everything created by the session.

URL that will use this file: taskerMAN/menu.php

3.1.20 RemoveMember.php

File Name: removeMember.php

Purpose: The purpose of this file is to remove a member from the table.

Functions Needed: We will need to loop through the database to select and input the data of the names of the users/members into a drop box located at the top. Once a name is selected, more information on that user such as email should be loaded. Which will need another loop to check for the data matching the 'ID' of the first person.

URL that will use this file: taskerMAN/removeMember.php

3.1.21 deleteMember.php

File Name: deleteMember.php

Purpose: The purpose of this page is to process the request sent from the members.php file, which will be to delete a member.

Functions Needed: Within this php file the functions I will use Post methods to pass data into this file which will be the userID and ID of the file. I will also use two sql commands that will firstly delete the selected user, based on the ID given. Secondly I use an update function to allocate the person that is being deleted task to another person. I will use php functions amongst these commands to make the connection and also terminate the connection. The last function I will be using is header, this will take the user of the site to another page, which in this case is “members.php”.

URL that will use this file: taskerMAN/members.php

Pseudo:

```
Define id from POST value

Query Database Delete from members where the id equals id variable


Define newname as Picture location from id


If newname exists Then
    Delete newname Picture file
End If


Redirect to “members.php”
```

3.1.22 index.php

File Name: index.php

Purpose: The main purpose for the index page is to have the user login to the site. Having a login page is essential when a website such as this can hold sensitive data, thus eliminating unauthorized access to the website. The secondary purpose of this page is to allow the user to download our TaskerCLI application which is to be run in Java.

Functions Needed: There will be no php functions used on this page.

URL that will use this file: taskerMAN/index.php

3.1.23 Test.php

File Name: Test.php

Purpose: The purpose of this file is to create a table if not there.

Functions Needed: These only functions needed will be the connecting functions used in the SQL statements.

URL that will use this file: test.php

3.1.24 Homestyles.css

File Name: Homestyles.css

Purpose: The main purpose of this file is to create a standard within our website, that will allow us to set styles of certain characteristics. The main use of this page is for the home page, it sets the style for the boxes and hyperlinks.

Functions Needed: There is no php functions used on this page.

URL that will use this file: taskerMAN/index.php

3.1.25 Membersstyles.css

File Name: Membersstyles.css

Purpose: The main purpose of this file, is to create a standard for this webpage which allows the users to set styles for certain characteristics. This file will mainly be used on the members.php file to create the style of the page.

Functions Needed: There is no php functions used on this page.

URL that will use this file: members.php

3.1.26 Styles.css

File Name: Styles.css

Purpose: The main purpose of this file, is to create a standard for this webpage which allows the users to set styles for certain characteristics. This CSS file, will be used in all other web pages. This file will set all of the body styles, navigation styles, buttons and header styles.

Functions Needed: There is no php functions used on this page

URL that will use this file: all of the webpages.

3.1.27 tasksStyles.css

File Name: tasksStyles.css

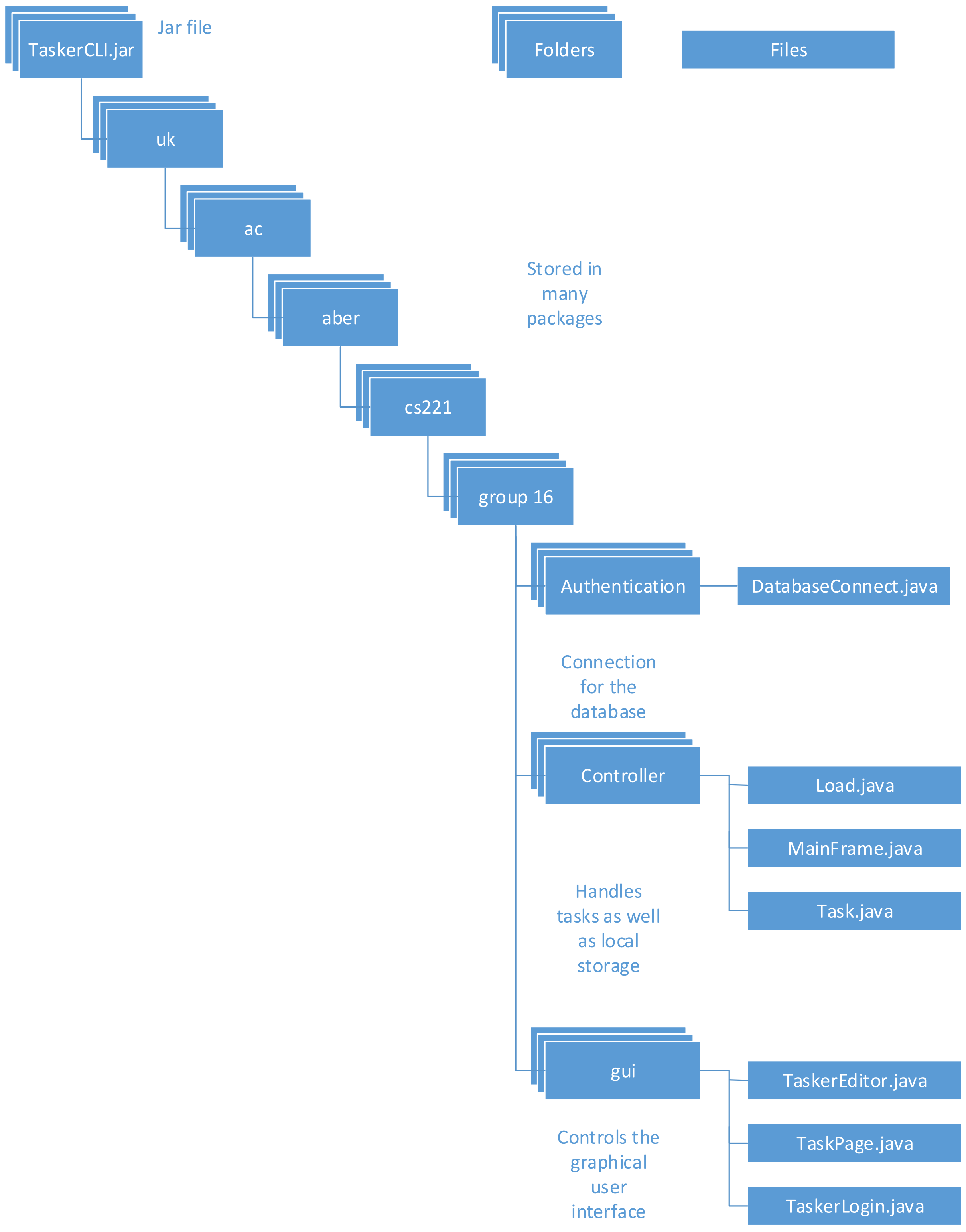
Purpose: The main purpose of this file, is to create a standard for this webpage which allows the users to set styles for certain characteristics. This CSS file will be used in the task php files (createTask, viewTasks, etc..) and will be used to create the styles of the buttons and text boxes.

Functions Needed: There is no php functions used on this page

URL that will use this file: createTask.php, editTask.php, viewTask.php, viewTasks.php

3.2 TaskerCLI Files

This is what the file store looks like:



As mentioned in the design speciation the Java application uses the Model-view-controller, to organise the logical of the program into discrete packages called authentication, controller and gui. The Java application has seven modules that are called from the class TaskerLogin. Please refer to the Design specification for the final system [2], for the main design and structure of the program.

3.2.1 DatabaseConnect.java

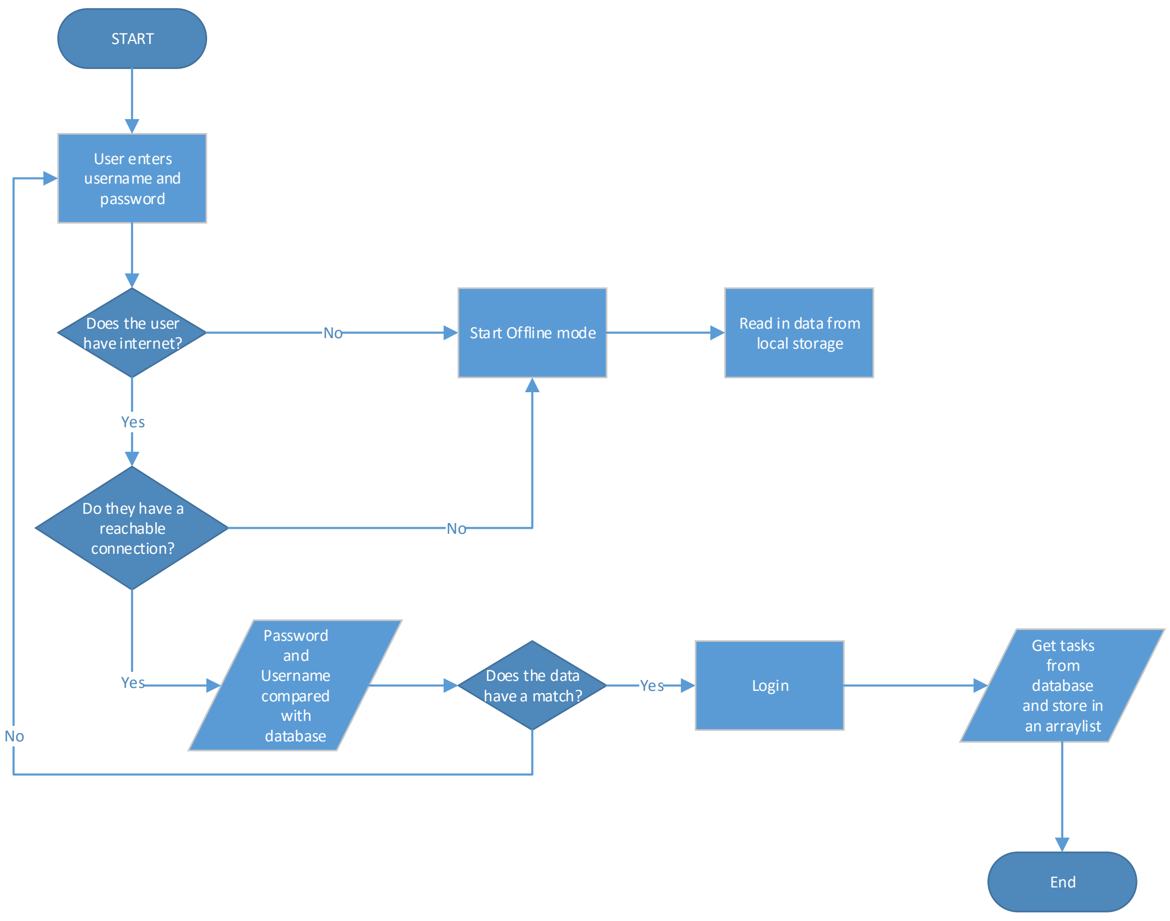
File Name: DatabaseConnect.java

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:


START
User enters username and password
Does the user have internet?
No
Start Offline mode
Read in data from local storage
Yes
Do they have a reachable connection?
No
Yes
Password and Username compared with database
Does the data have a match?
Yes
Login
Get tasks from database and store in an arraylist
End
No

Methods:

getUserName() – returns the username needed to login to the MySQL database.

checkForInterfaces() – checks for a working networking card that is used to connect to the network. It will return a Boolean value of either true or false.

isReachable() – returns a Boolean value of either true or false. Its parameters are:

- Addr – the address that you want to check is available
- openPort – the port you want to access
- timeOutMillis – the timeout before it stops checking if the connection is reachable.

haveInternet() – Checks to see whether there is an internet connection. It returns a Boolean value of either true or false.

Hashing() – this method is used for hashed passwords. The parameters it takes in, is the password entered and returns the un-hashed password.

Login() – This is used to log the user into the application and the database server. The parameters used is the email and password entered into the form, it returns a Boolean value of either true or false.

Connect() – this method creates the connection to the database, it returns a connection that is used to communicate with the database.

getTasks() – this method is used to get all the tasks from the server that the member will need. It saves all the tasks to an arrayList.

uploadAllTasks() – takes the arraylist as a parameter and uploads them to the database.

Disconnect() – disconnects the connection to the database.

Sync() – takes the arraylist as a parameter and return the updated task list. This will first download new tasks, if there are any, then will delete tasks that are deleted from the server and update the local changes.

checkForNewTasks() – The parameters used are the tasks arraylist and another arraylist which holds all the tasks that are on the server. The method will check if any new tasks on the server are not downloaded yet. It returns the updated task list.

RemoveDeletedTasks() – This method deletes any tasks that you might already have. Its parameters are the tasks arraylist and another arraylist that holds all the tasks that are on the server. Its returns the local tasks.

IsLoggedIn() – returns true if the user is logged in.

setLoggedIn() – takes in a Boolean value of whether the user is logged in or not. Sets the status of the login.

getDidItSync() – returns if the database has synced or not.

3.2.2 Load.java

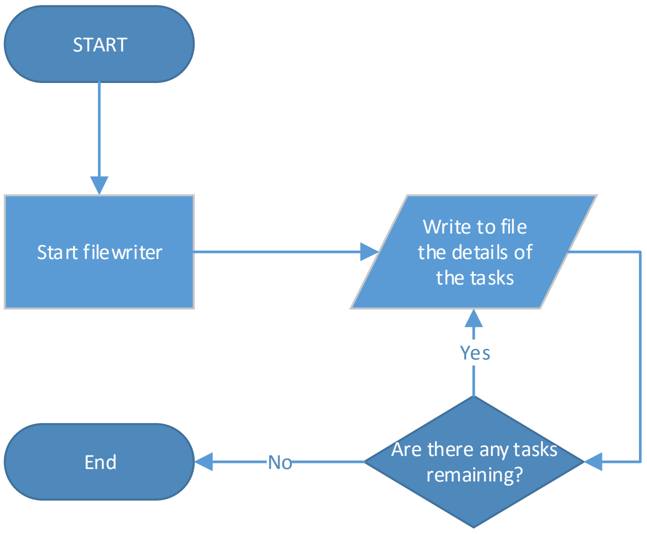
File Name: Load.java

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

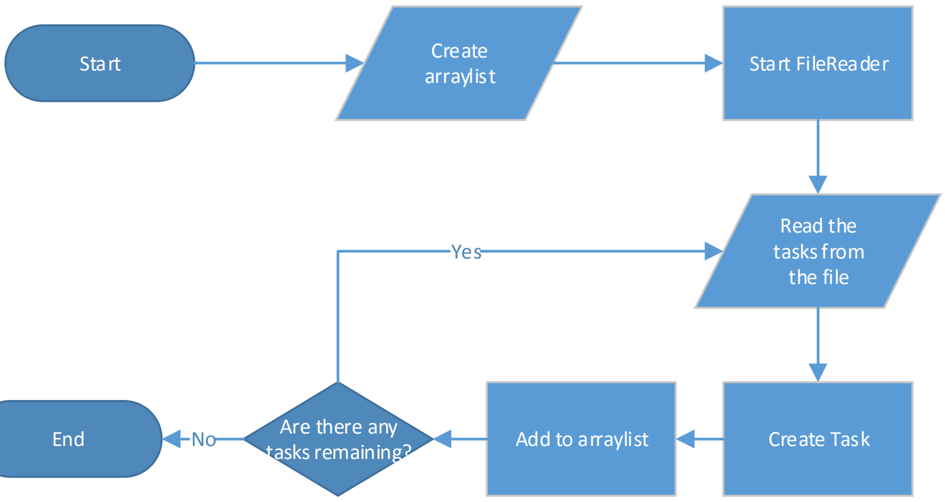
Dependency: DatabaseConnect.java

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:


START
Start filewriter
Write to file the details of the tasks
Yes
End
No
Are there any tasks remaining?

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



Methods:

Load() – creates the fold and the file needed to store the tasks locally.

getUserName() – gets the username and returns it.

Save() – Uses tasks and the username as parameters. It takes the arraylist of tasks and saves them to the correct location.

countLines() – counts the number of lines in the task description and is used to be able to store multiline descriptions. It takes the description as a parameter and returns the number of line used.

Load() – This method loads the tasks. It takes the parameters of the user and checks to see if they are logged in (Boolean value). It then returns the arraylist with all the tasks.

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.**

Methods:

Run() – Starts the GUI in a thread safe environment.

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.

Methods:

Task() – this is the constructor used to create the task object. It takes in the ID, username, the task information, end date, title and the start date.

Each of the objects variables mentioned above in the task object, has a getter and setter. This will just set or return the variable of the chosen object.

toString() – This returns and prints the task object in the chosen format.

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.

Methods:

TaskEditor() – this takes in the parameter of the task and the database connection. This method sets out the GUI for the window that will be used to edit the tasks.

3.2.7 TaskPage.java

File Name: TaskPage.java

Purpose: This class displays all the information of all tasks.

Dependency: TaskerLogin.java

How it works: 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.

Methods:

TaskPage() – this method takes in the database connection. It then sets up the GUI for the tasks page. This includes inner methods which are used to control the timings. These include refreshing the GUI every second and also syncing every 5 minutes.

deleteItemFromJList() – this method take an index of type int then removes that index from the model, it then gives default values to the JPanel.

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.

Methods:

TaskerLogin() – Sets up all the GUI of the login page.

Login() – checks whether the page needs to change depending on if the user has logged in or not.

Validate() – validates email with regular expression. Takes in a hex value and returns whether the hex is valid.

4.0 Algorithms

Most of the code used in both parts of the Tasker project are very straight forward and no significant algorithms have been used in either.

5.0 The Main Data Areas

The Java application, has two main data structures, the ArrayList that stores tasks in memory while the Java application is running. Then a raw text file stores the ArrayList information, for the use in offline mode.

The database holds most of the data that is needed. The database uses MySQLThis is how the database is formatted (copied from the design specification):

tasks { TaskID, StartDate, DateOfCompletion, TitleOfTask, MemberAllocated, Status, Comments }

Column	Description
TaskID	Every task that is created has a unique number to identify the task. This is the primary key. It also helps with the editing on the TaskerMAN, as “php?id=5” could be used instead of “php?=john20”
StartDate	This is to let the user know when they would have received the task, it helps the user to organise their time.
DateOfCompletion	This is for when the task needs to be completed by, effectively a deadline field.
TitleOfTask	To allow the user to know which task is which and to easily navigate through the list.
MemberAllocated	This is to allocate the task to a specific person. This is then checked with TaskerCLI when reading in the data.
Status	Status can only be three different states, these being allocated, completed or abandoned. This is for the managers to view as well as the TaskerCLI to read in only the relevant data.
Comments	This is just for additional comments for the task.

members { id, name, email, password }

Column	Description
Id	Every member needs to be unique, this is the primary key. It helps with the editing on the TaskerMAN web page, as “php?id=5” could be used instead of “php?=task50”.
Name	This is a way of identifying users for the user, we don’t expect the managers to remember everyone’s id.
Email	Email address to login with for both applications, two different accounts can’t have the same email address, effectively a username.
Password	Optional feature that we have decided to add, to secure accounts and keep them personal.
Admin	This column is to let the TaskerMAN know which users are actually managers, as managers are the only people that can log in and allocate tasks. This will be a Boolean field.

6.0 Files

Tasks.txt is created with TaskerCLI to store information that is read in from the database. It uses this file to store the current state of the database. If any changes are made then this is edited and saved. Once the application is closed with a valid connection it will update the database. This is the format that the file stores its information:

- Number of tasks that need to be read in/saved.
- The user that has made the changes.
- The ID of the user that the task was allocated to.
- The task title.
- The task information.
- The start date.
- The deadline.
- The status.

It saves it in the OS’s home directory.

7.0 Interfaces

The only interface that is used is in the checkForInterfaces() method within the DatabaseConnect class. This checks to see whether there is an available network card to use to connect to the internet.

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.

One improvement would that has been suggest is to correct all bugs that the software might hold. This would need through testing, since this is very time consuming, we did not fully test the program as much as we would have liked to. One bug nearly always lead to another. All bugs that we found were recorded and fixed, but this doesn’t mean that the program is bug free.

Another improvement would be matching all the requirements that were needed for the program to be fully accepted. The requirements specification would need to be thoroughly read through again and make note of each requirement that wasn’t implemented into the two programs that were created [1]. During the acceptance testing a lot of the tests failed and we didn’t get a fully accepted project. The filtering and sorting could be implemented fully as this was a requirement that we missed.

There were problems with the web application, some browsers weren’t as fully compatible with HTML5 as we had hoped. So the improvement would be that the web application fully worked with the most popular browsers. One example of this problem was that when using Google Chrome on mac OSX, it would have problems with the layout and sometimes the calendar function wouldn’t work. One improvement would be to check if the user has the latest version of the browser, this could then link them to an update.

The website could be improved by adding new features to make the accessibility easier for the user, by using search, filter and sorting. This was in the requirements specification, and should have been implemented [1]. This would be useful as the user could look at a specific data list instead of ever task that is to be completed.

An improvement that could be made is the synchronisation could be fully functional on TaskerCLI. This was working on all computers that we tested on, minor changes were made before submission of the project and this caused the synchronisation to fail during the acceptance testing. This could easily be fixed.

One of the improvements that could be suggested is checking the software for bugs more thoroughly, since it was one of the time consuming problem while doing the group project as one bug error lead to another error which meant that more time was taken than planned in order to fix the bugs and make the program work. To tackle if the same problem occurred in the future, all of the bugs should be recorded in order to make sure that it can be dealt with early when working on the program rather than later.

9.0 Things to watch for when making changes

If someone were to change the database then this would stop both the TaskerMAN and TaskerCLI to stop working. Both applications rely on the database in order to store and retrieve information. This is the central part of the applications that need to be carefully changed. If fields of the tables are deleted or changed then this could stop things from working at all.

If the location of the database was changed then this would need to be updated on both the applications, this would be the connect.php file and also the java file called DatabaseConnect. As the location of the databases are hardcoded, this would need to be changed.

Minor changes to the task class could potentially ruin the upload, download and synchronisation when editing tasks. This is the object class that is used, so it needs to be carefully maintained.

Most of the PHP files are quite sensitive as they do not use functions. They just use include, so each file needs to be working 100% for the whole program to work fully.

Above are the main areas that need to be taken care when making changes, but the whole project needs to be carefully changed, there isn't just the one area, its all of the code that is sensitive.

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

The java application is a self-contained jar file, which connects to the database, due to this it only has the source code files to be concerned about. To rebuild the java application, the java source code would need to be recompiled into a jar executable. To test if the system is working, use the provided Junit tests. Only a raw text file is accepted, any other file type would not be useable and would need rebuilding.

The TaskerMAN is a collection of PHP and CSS files. To rebuild this the files just need to be placed onto a hosting server. Permissions should not be changed, but if one part is not working then they may need to be looked at.

For testing the applications please use the test specification that has been provided, this should be followed and if any new features are introduced then new tests should be created.

References

[1] – QA Document SE-QA-RS – Requirements Specification 1.2, N.W Hardy.

[2] – SE_16_DS_02 - Design Specification for the Final System 1.6

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	2016-02-12	Ready for release.	Robert Mouncer –rdm10
1.3	N/A	2016-02-12	Ready for Review	Robert Mouncer –rdm10