Group 16

Design Specification for the Final System

Author: Richard Price-Jones, Archie Strange, Greg Sharpe, Rhodri Pearce, Emil Ramsdal, Robert Mouncer

Config Ref: SE_16_DS_02

Date: 26/11/2015

Version: 1.6

Status: Draft

Department of Computer Science Aberystwyth University Aberystwyth Ceredigion SY23 3DB

Contents

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to describe and illustrate the specification for the design of our system. It contains all the descriptions that will be necessary for the implementation phase of the project.

1.2 Scope

This document specifies the high level for our system. It includes designs for interfaces, software structure, components and data. It describes how our applications will look for the user and how they will interact with the each other.

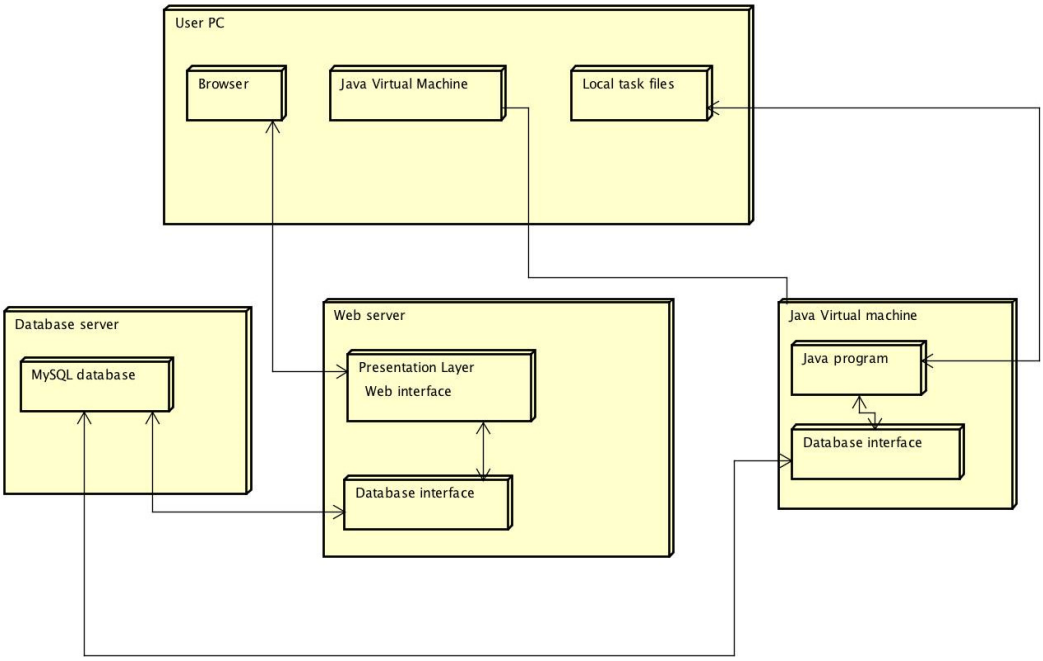
1.3 Objectives

The objective of this document is to provide a framework design that will be used throughout the entire project. It will ensure that the applications have the all the functionality requirements set out by the client [1]. It will also be crucial during the implementation phase as it shows how everything should work and it will be used as a plan.

2.0 Deployment Description

2.1 Applications in the system

The deployment of this software requires the user's computer system to have a modern web browser and the installation of the Java virtual machine. A Web server is required to support PHP which is used to communicate with the database. The database server is required to be running MySQL, the database is also required to allow remote login.



2.2 Applications Interactions

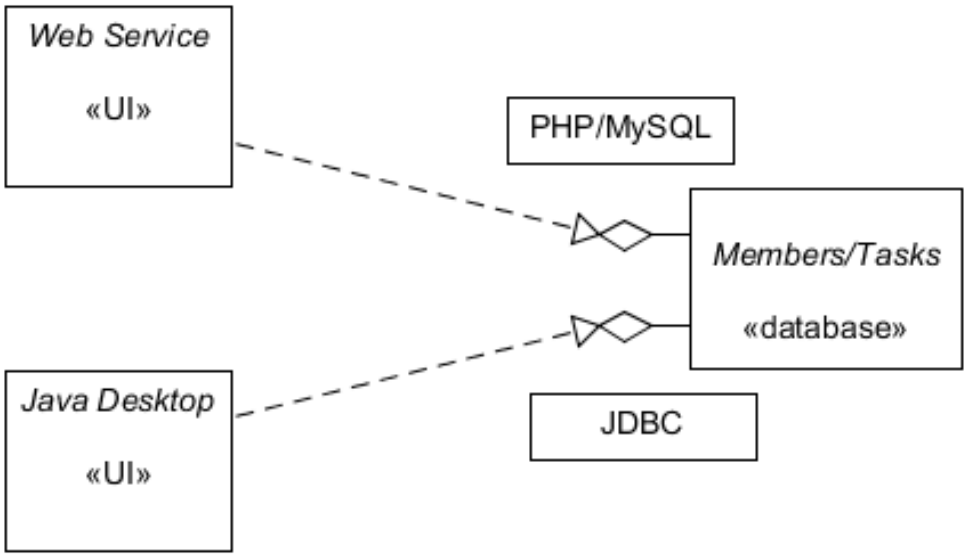
The Web service will use PHP on the server to connect to the database, firstly we must establish a connection within the PHP script. To connect to the database using MySQL we must first use the function mysql_connect, along with the username, password and hostname of the database. Then once connected we may begin to run queries, the function used to perform these queries is named mysql_query(). Lastly, we will need to close the connection, although this isn't necessary as PHP automatically closes the connection when the script ends. By using the mysql_close() function we will close the connection.

JDBC is a driver which allows the Java Application connect to a data source, in our case it's a MySQL database. It will allow the Java Application to send and update query statements and

process the results. JDBC will access our remote server using the Internet's file addressing scheme and a file name our on server, which in our case will be our database name.

The communication protocol we will be using within both JDBC and PHP is HTTP. HTTP meaning Hyper Text Transfer Protocol will enable the PHP script and the JDBC to communicate with the database by firstly declaring a port on which our database will be accessible. In this project will we have a connecting PHP page which will include the database address, username and password. This PHP page will be called many times.

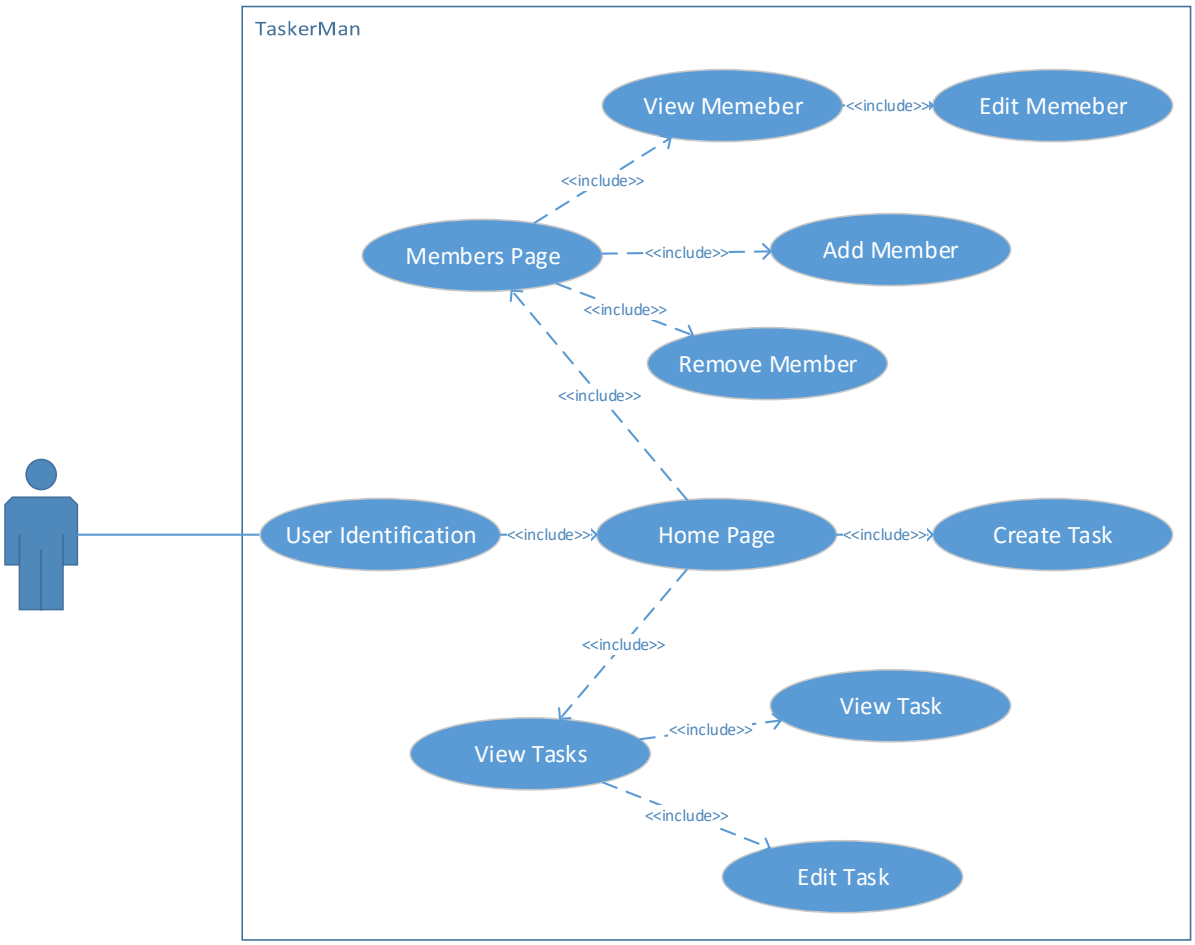
Below is a component diagram illustrating everything that is mentioned above.



3.0 Interaction Design

3.1 Use Case Diagrams

Below is the Use Case Diagram for the Web Application Tasker Man.



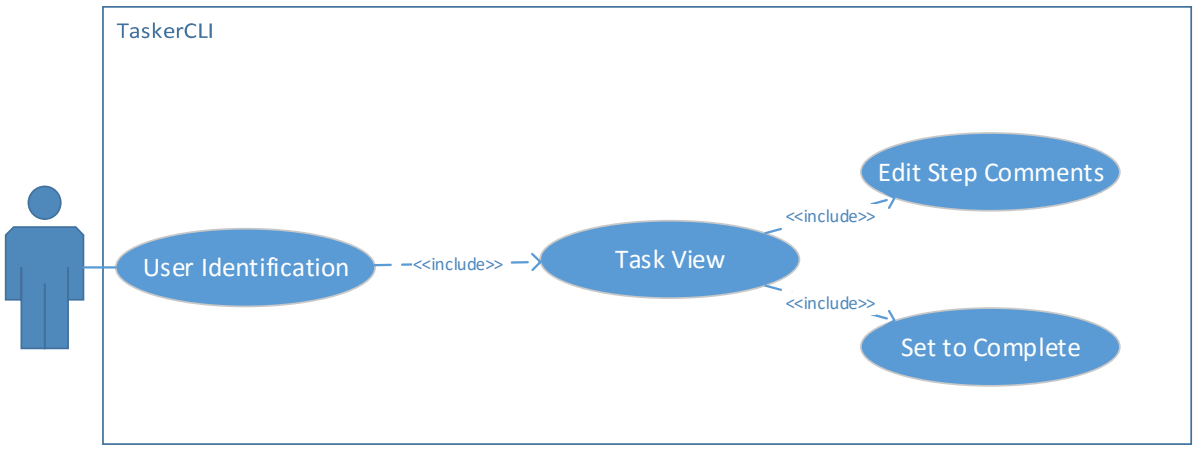
The name of the system (TaskerMAN), is displayed on the top left within the System boundary.The Actor in this case is the Manager of the project, which can be seen to the left of the use case, just outside the system boundary.

The system boundary contains all of the functionality or services provided by the system.

Associations within the use case diagram are used to display the connections between the pages of the website. From this we can see that the user starts at the home page, from there can visit any one of three separate pages (Members, Create task and View Task). From these sections, the user can travel further into the website.

- Through Members, the user can reach ‘Members Info’, and from there, ‘Edit Members Info’.
- From the View Tasks page, the user can then visit one of two options. ‘View Task’ and ‘Edit Task’.

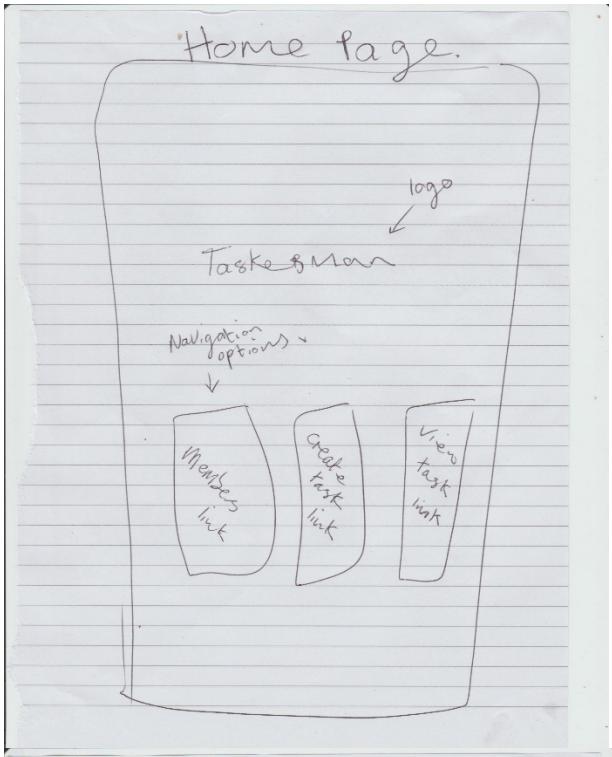
Below is the use case diagram for the user application Tasker CLI.

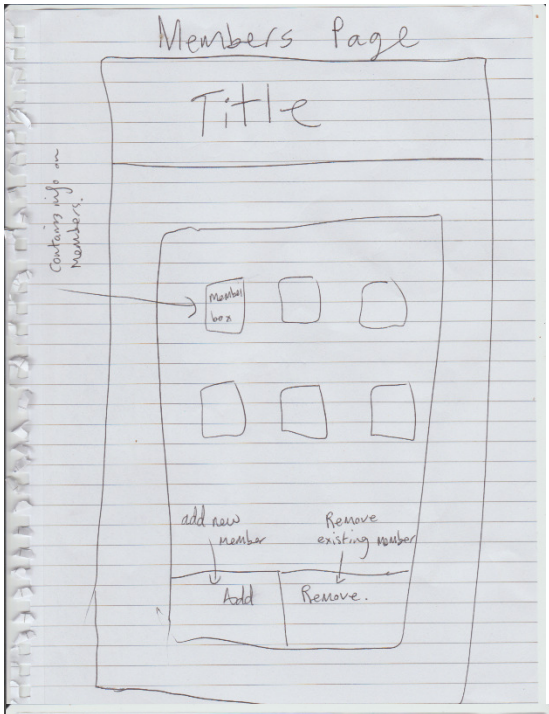


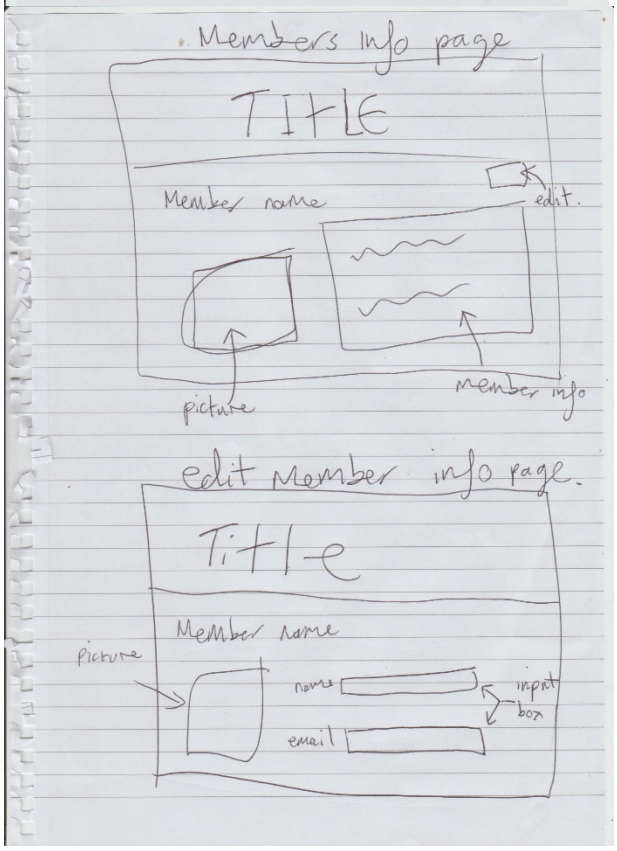
The associations are used in the use case diagram are used to display how the use interacts with the application. You can view all the functionality of the program. So we can see that the application will provides: user identification, the ability to view tasks, to edit the comment about the tasks and finally to able to set the user's assigned task to completed.

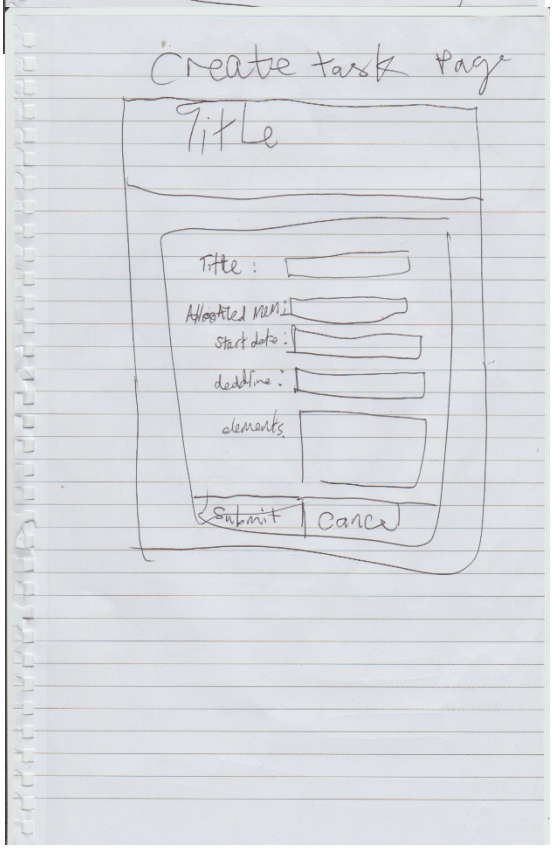
3.2 User Interface Design

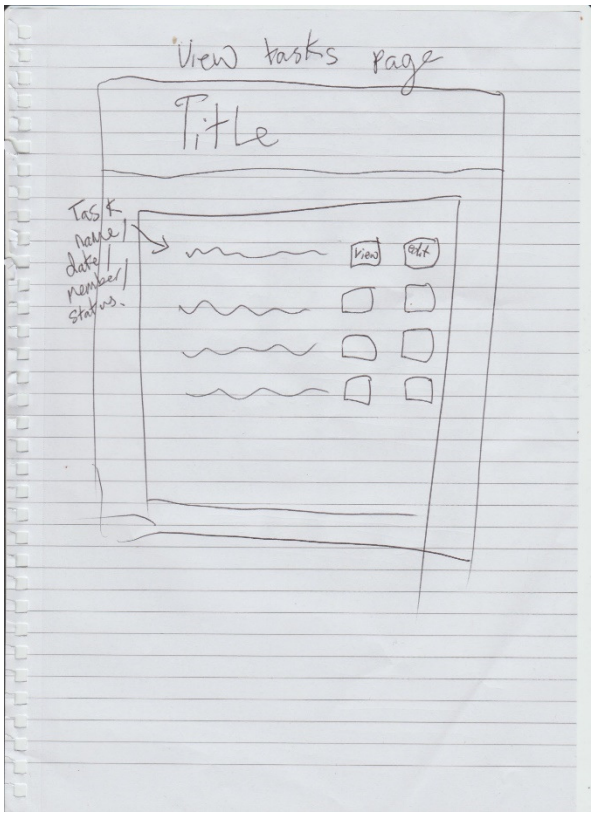
3.2.1 Rough TaskerMAN Designs

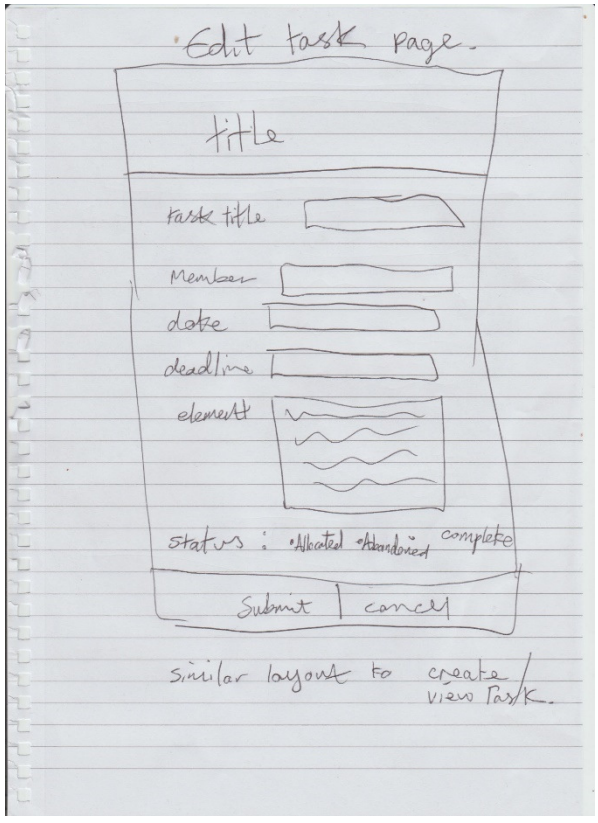












These designs are a rough representation of the layout of TaskerMAN. These will be used to help create the design of the website.

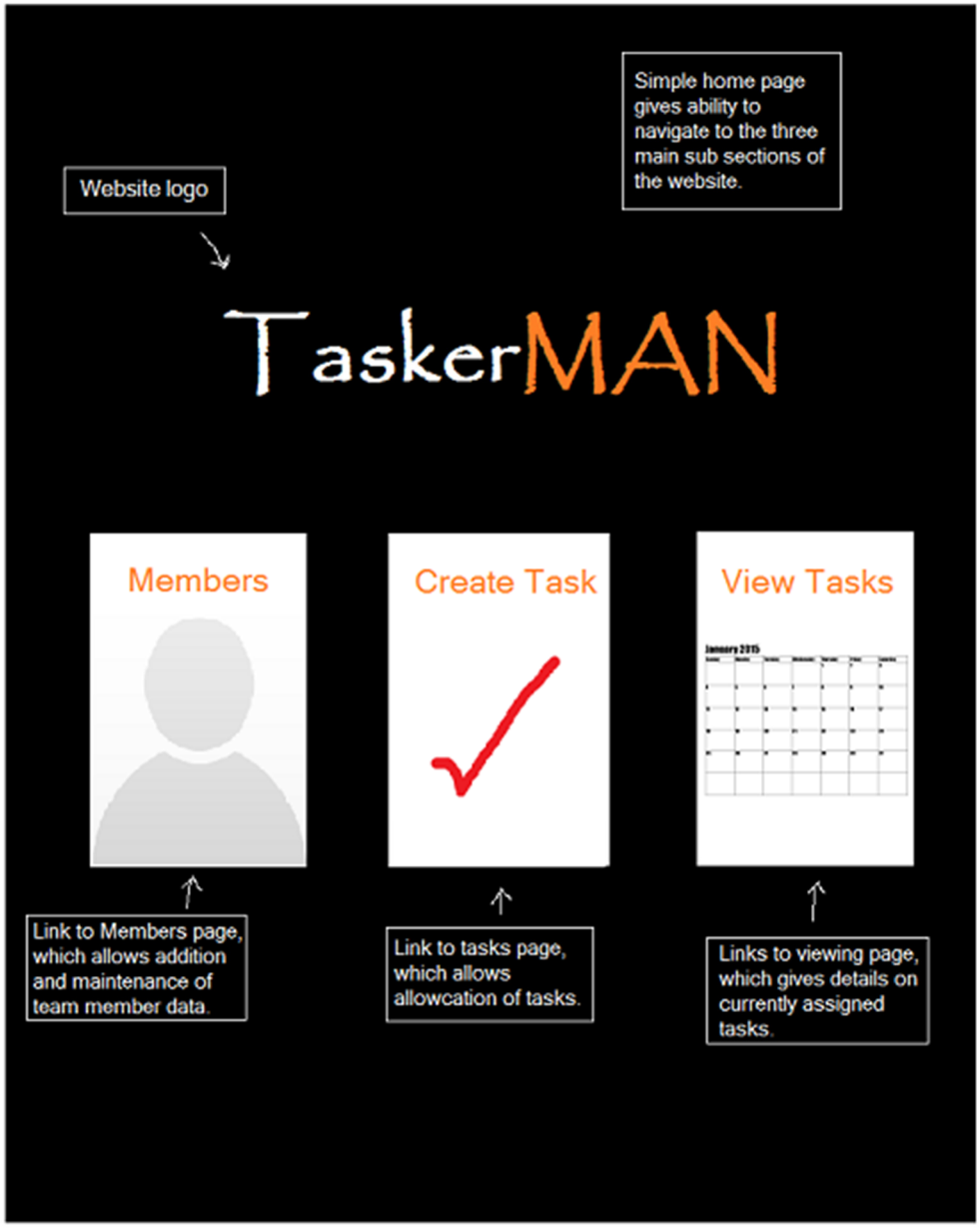
3.2.1a Template

The header of the website will not change throughout the site, and can be found on every page, with the exception of the home page where is does not exist, though the logo will still appear. This header will give easy access to all of the main sub sections of the site, and also the home page. The navigation is clearly defined, while clicking on the website logo will redirect the user back to the home page.

3.2.2 Home Page

For the home page we've designed a simple page that gives the user the choice to navigate to any of the three main sub sections of the website. Clicking on any of the three boxes will redirect the user to the selected page. The URL for this page will be:

http://users.aber.ac.uk/abc123/ taskerMAN/home.php

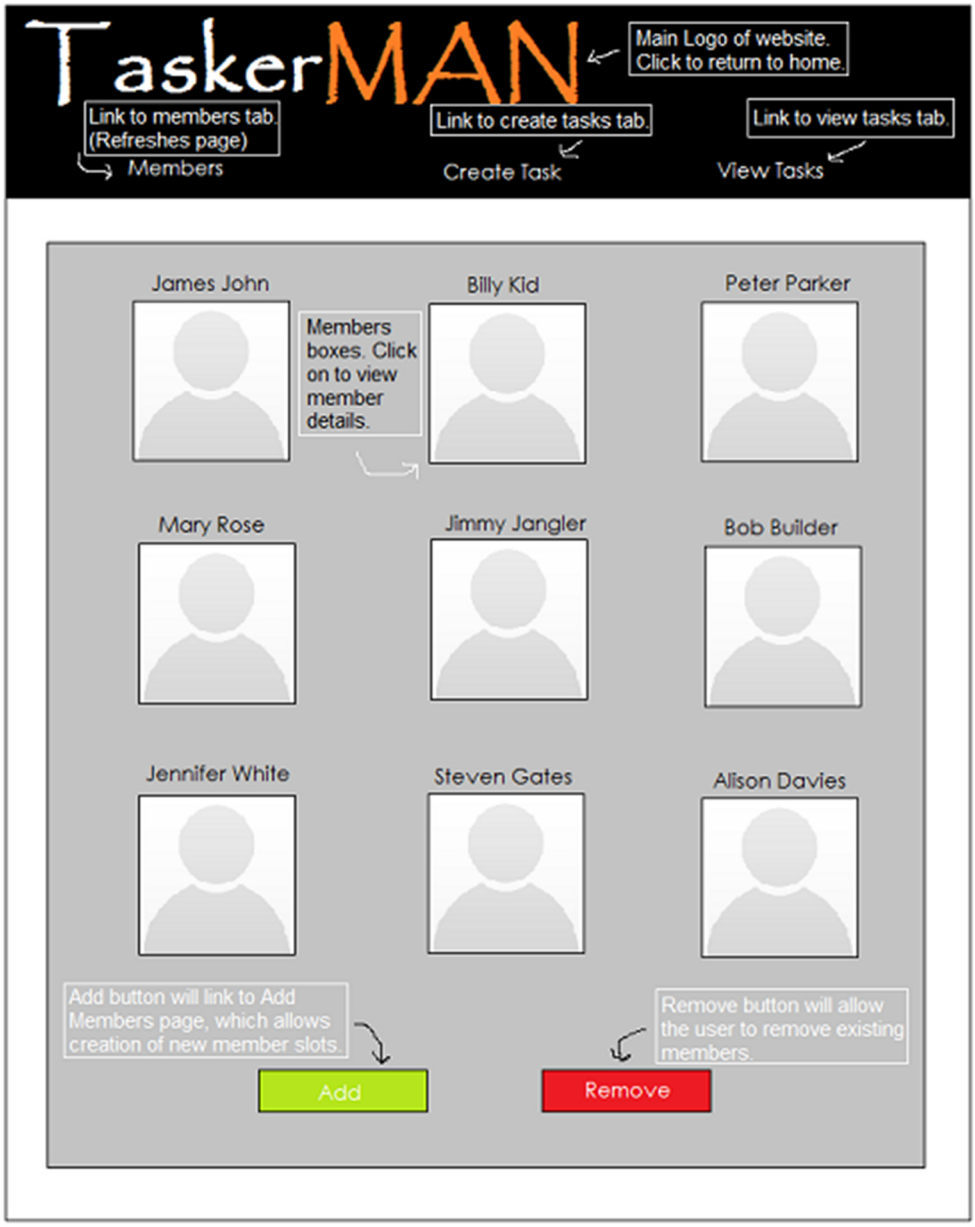


3.2.3 Members Page

The member's page contains all of the existing members. By clicking on any of the members portraits, the user will be redirected to the member's information page, which holds information on the selected user.

On this page there is the option to add or remove members. Clicking the 'Add' button will redirect to the 'Add Members Page', which gives the user the ability to create a new user with personalised information, while clicking on the 'Remove' button will give the user the ability to then click on the desired profile for deletion, without the hassle of moving to another page.
The URL for this page will be:

http://users.aber.ac.uk/abc123/ taskerMAN/members.php



3.2.4 Members Information Page

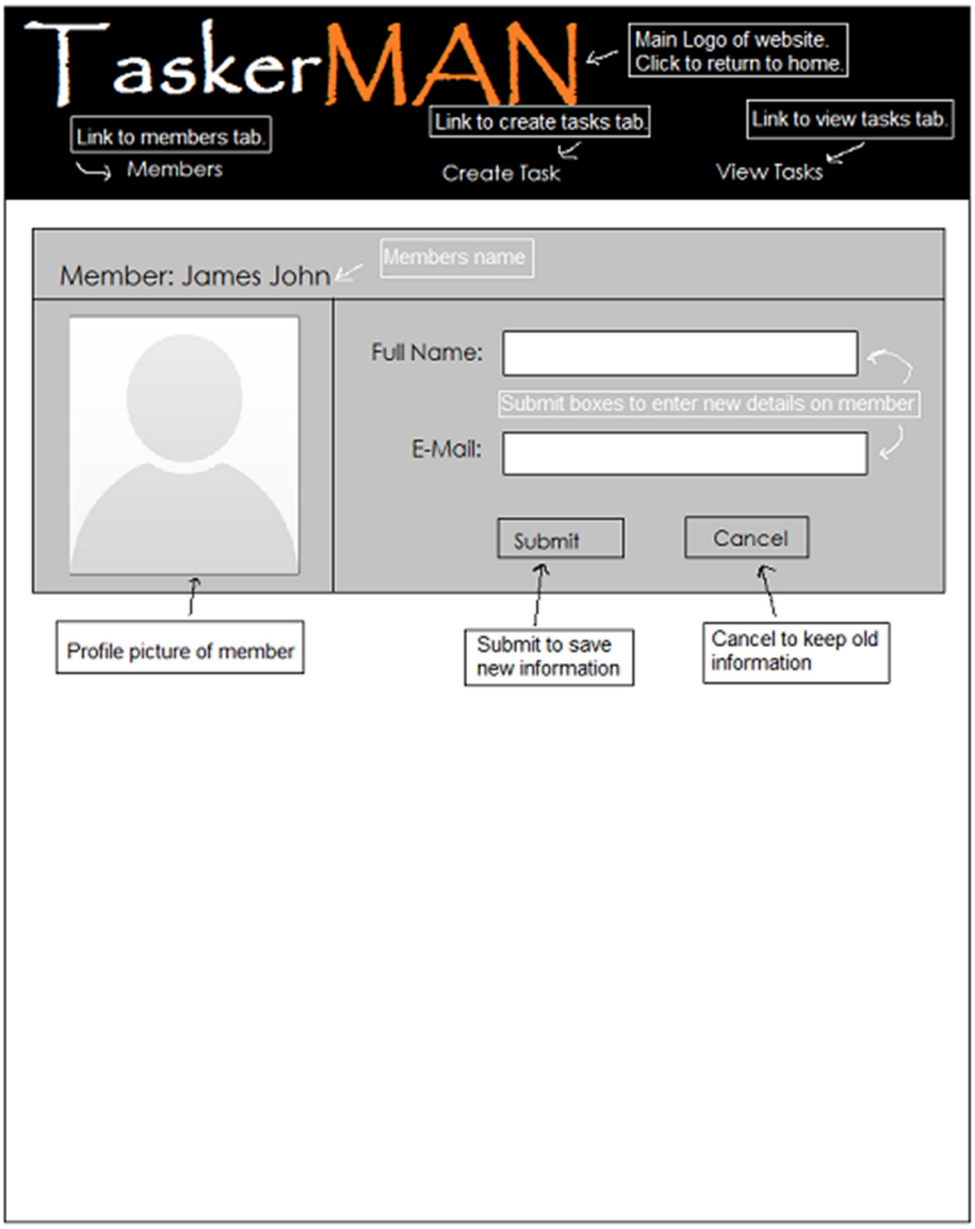
The information page contains the required details of the member. The edit button on the top right of the users information is a link to the 'Edit Members Information Page', which gives the user the ability to the existing information on the member.
The URL for this page will be:

http://users.aber.ac.uk/abc123/ taskerMAN/membersinfo.php?id=5

3.2.5 Edit Members Information Page

The edit information page is used to change the member's current details, which can be saved by submitting, or restored to its original state prior to editing by pressing the Cancel button.
The URL for this page will be:

http://users.aber.ac.uk/abc123/ taskerMAN/membersedit.php?key=5

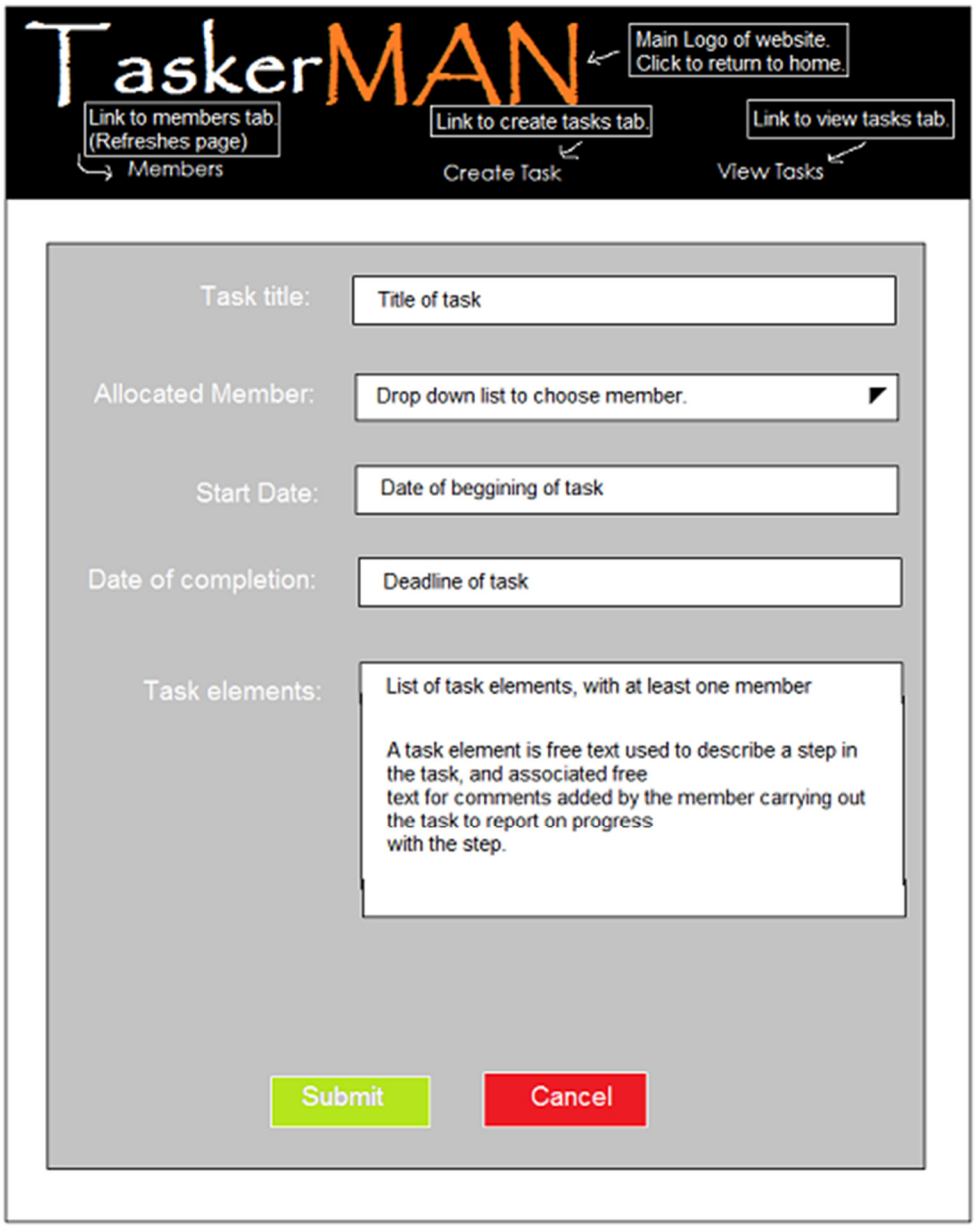


3.2.6 Create Task Page

The create task page allows the user to create tasks.

To create a task the site requires information to be submitted by the user: The member to which it is being assigned and various task details (Title of the task, Starting date, Deadline, Task elements). On clicking the drop down list, all available members will be displayed ready for the user to choose the desired member. On submission the task will automatically be set as "Active". Any mistakes made whilst creating the task can be altered through the 'Edit Task Page'.
The URL for this page will be:

http://users.aber.ac.uk/abc123/ taskerMAN/createtask.php

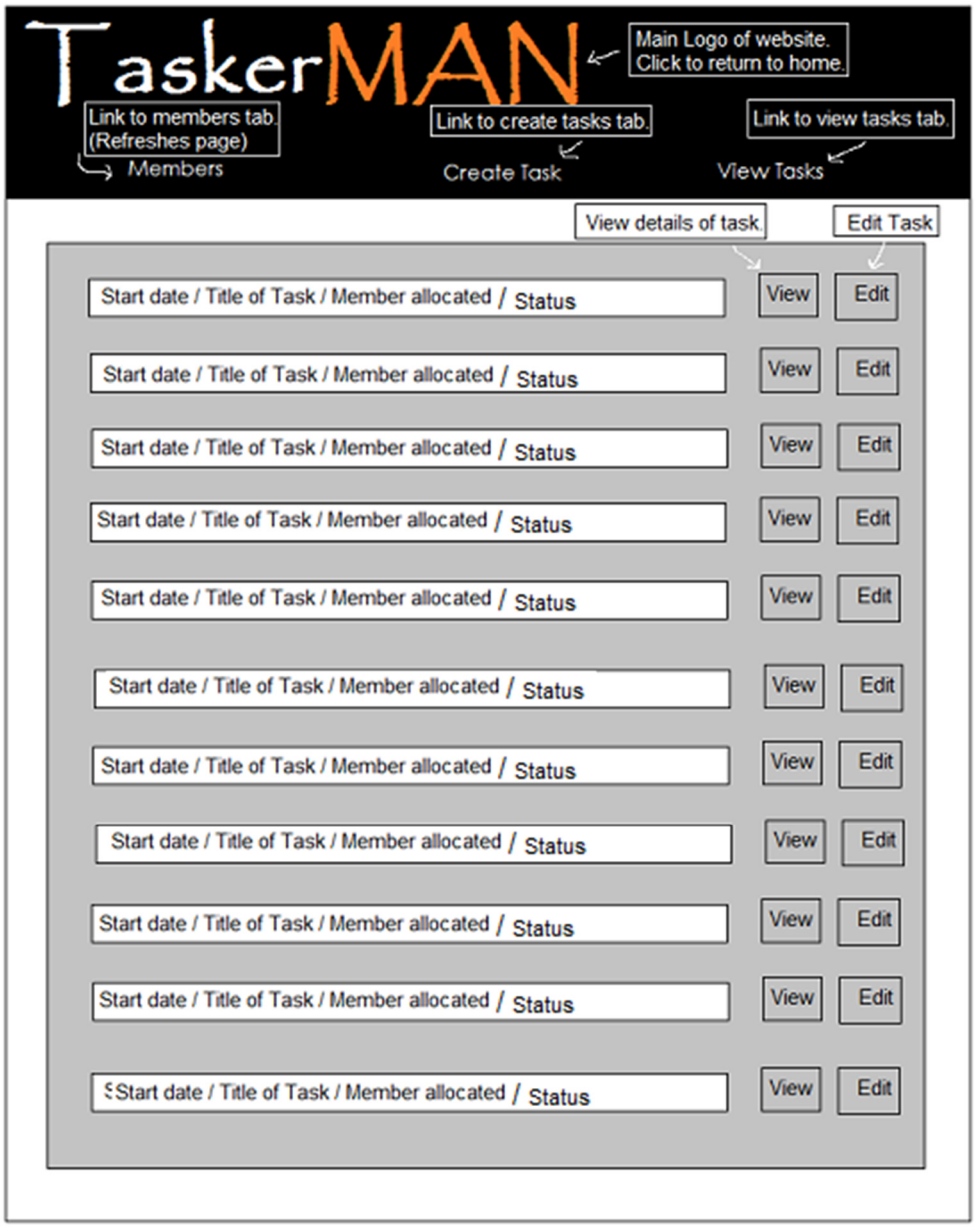


3.2.7 View Tasks Page

The view tasks page displays all the tasks that have been assigned, whether they be active, abandoned or complete. To view these tasks in greater detail simply click the view button adjacent to the description. The Edit button will redirect the user to the Edit Task page.

This list should be sorted by expected completion date. It should be possible to filter this list by task status and/or allocated team member.
The URL for this page will be:

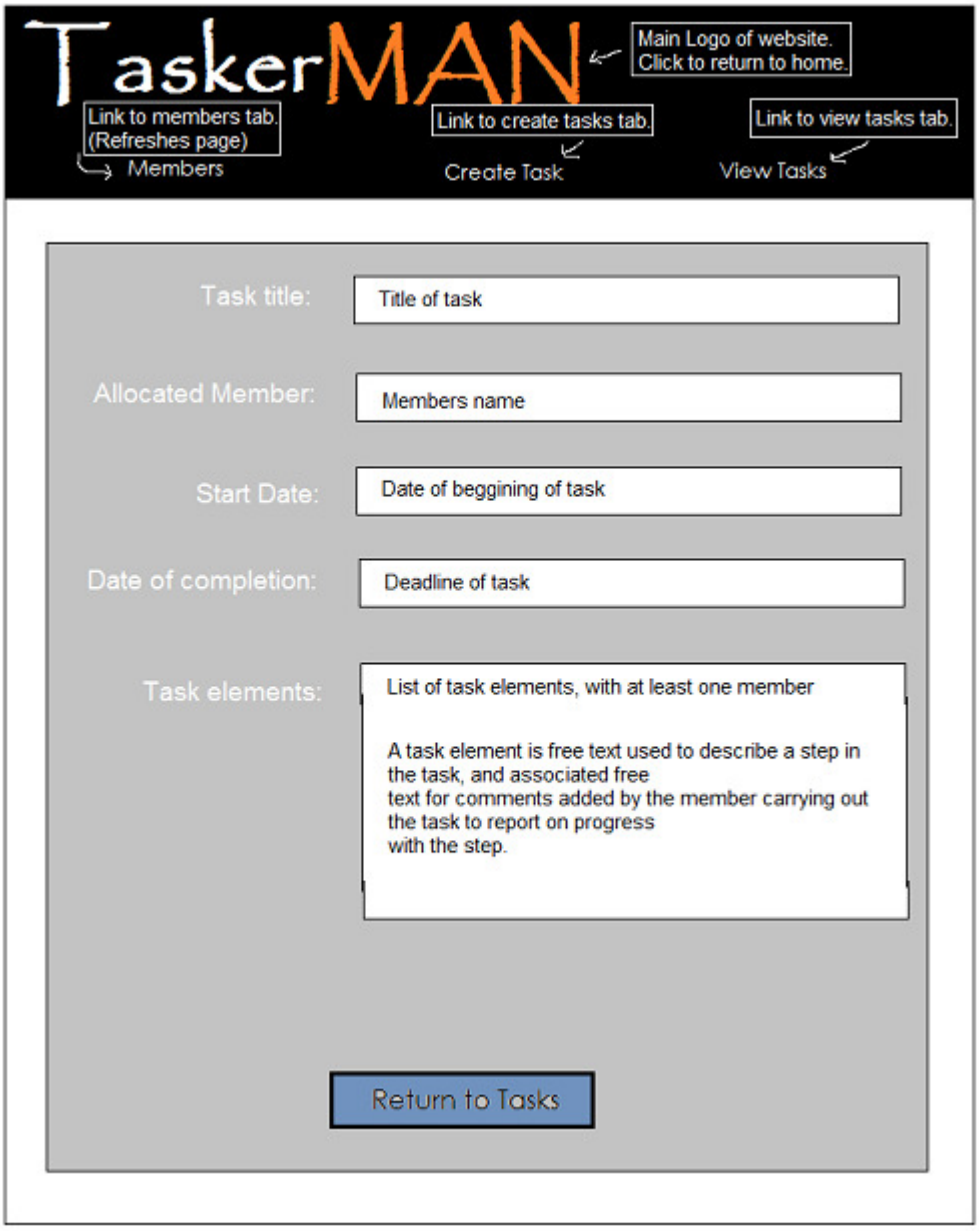
http://users.aber.ac.uk/abc123/ taskerMAN/viewtasks.php



3.2.8 View Task Page

The view task page simply allows the user to view a task in greater detail. Once the user is ready to leave the page, click the "Return to Tasks" button to return to the 'View Tasks' page.
The URL for this page will be:

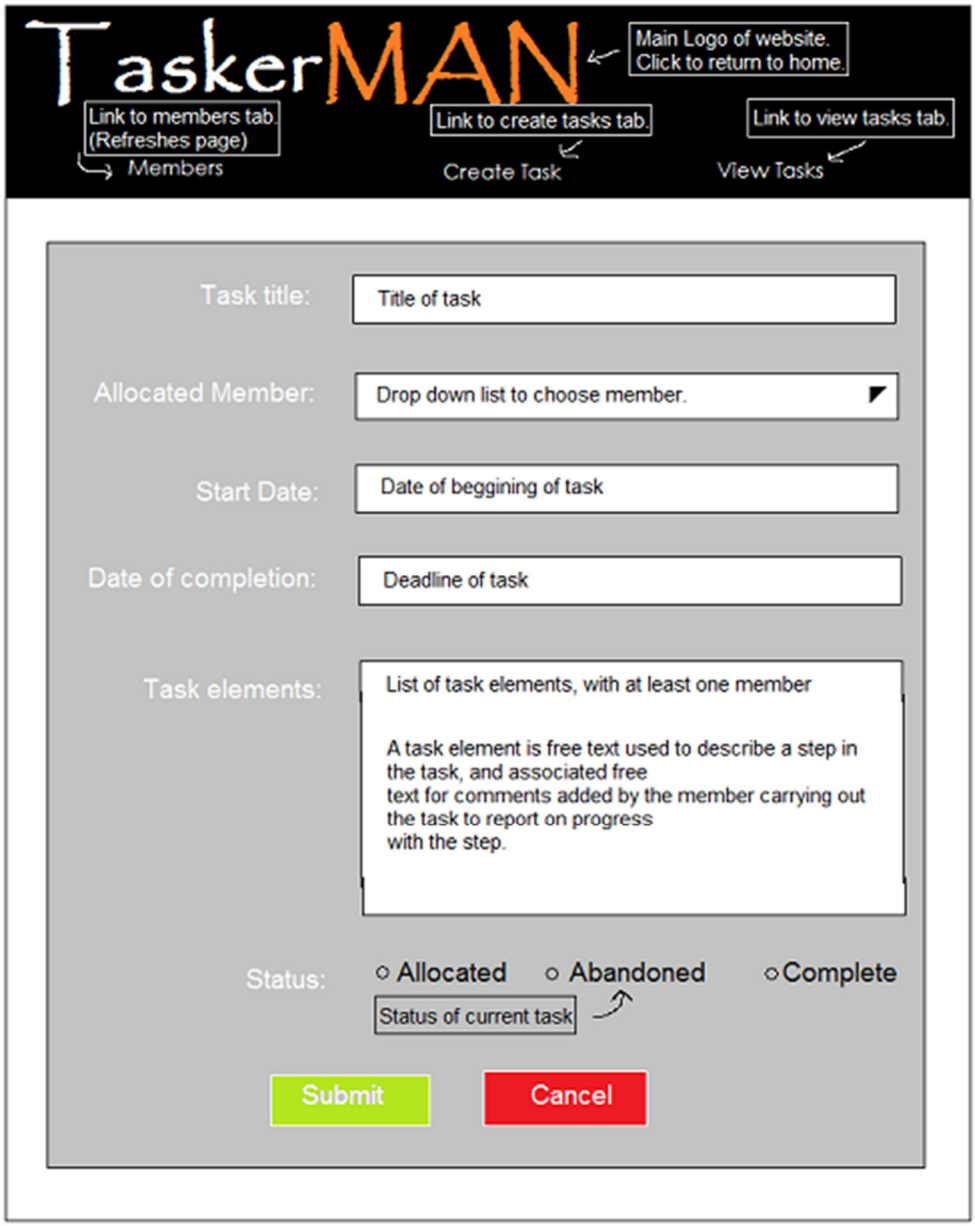
http://users.aber.ac.uk/djt/taskerMAN/viewTask.php?Taskid=52



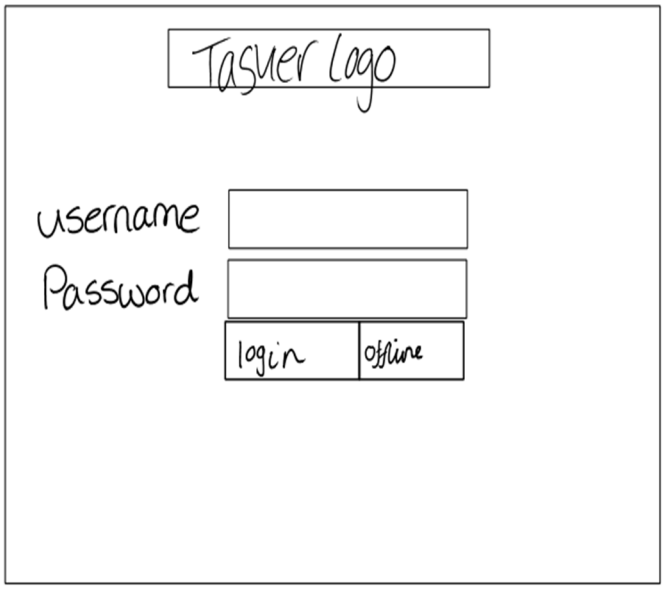
3.2.9 Edit Task Page

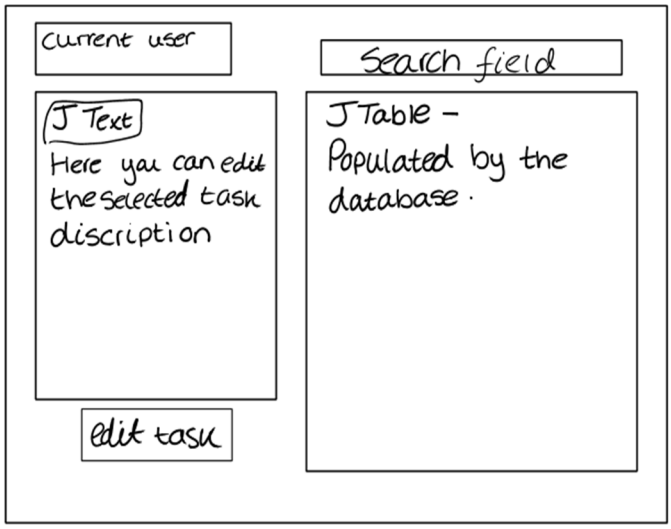
The edit task page will give all the same options as was had when creating the task originally, with the addition allowing the user to edit the tasks current status. From here the user will also be able to change the member to whom the task is allocated. The 'Submit' button will save any changes made to the task, while the 'Cancel' button will restore the original information prior to editing.
The URL for this page will be:

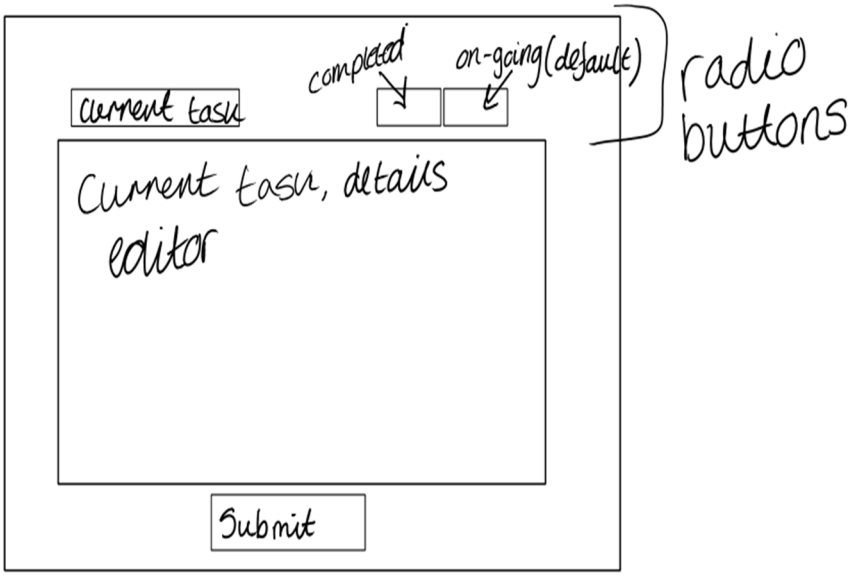
http://users.aber.ac.uk/djt/taskerMAN/editTask.php?TaskID=3



3.2.10 Rough TaskerCLI Designs



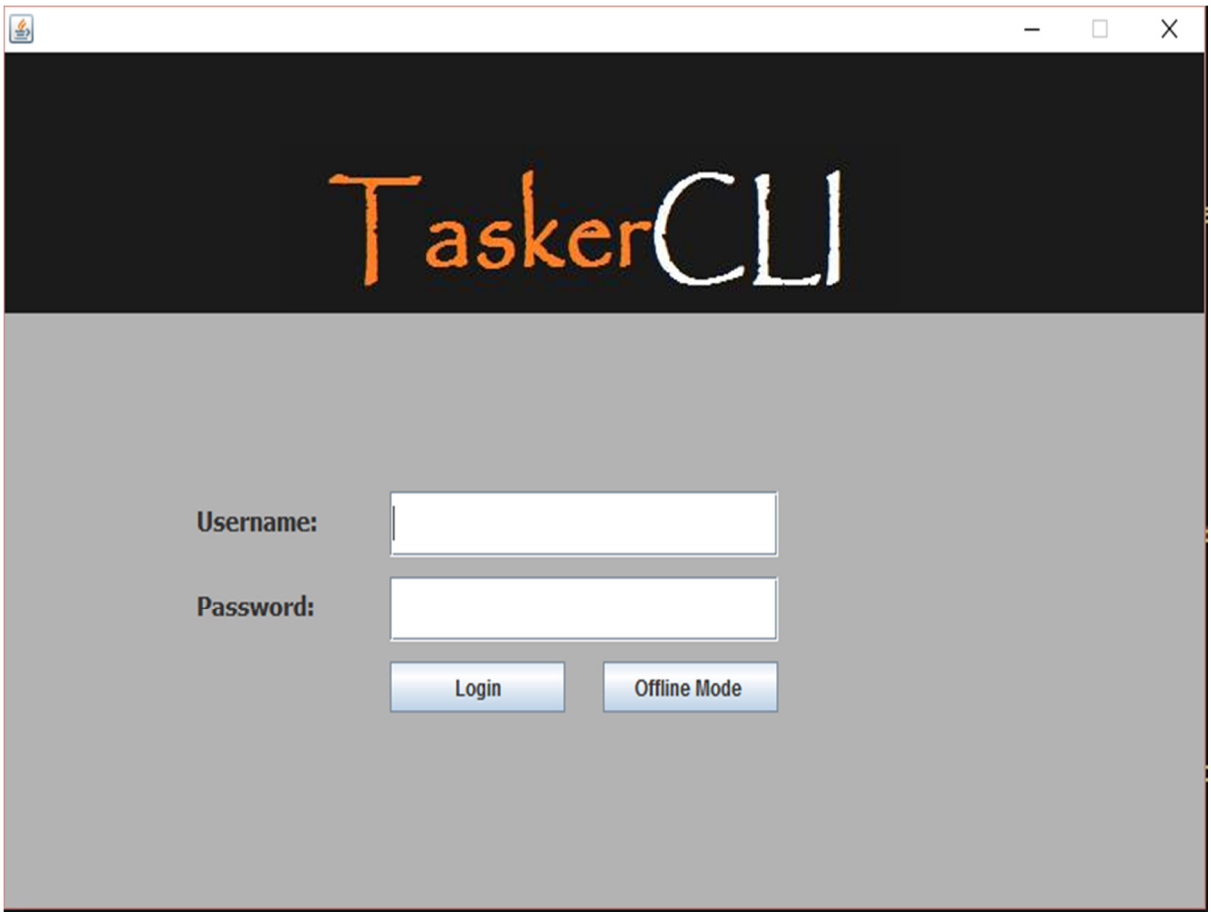




Below are the final designs for the user interface on the user application Tasker CLI

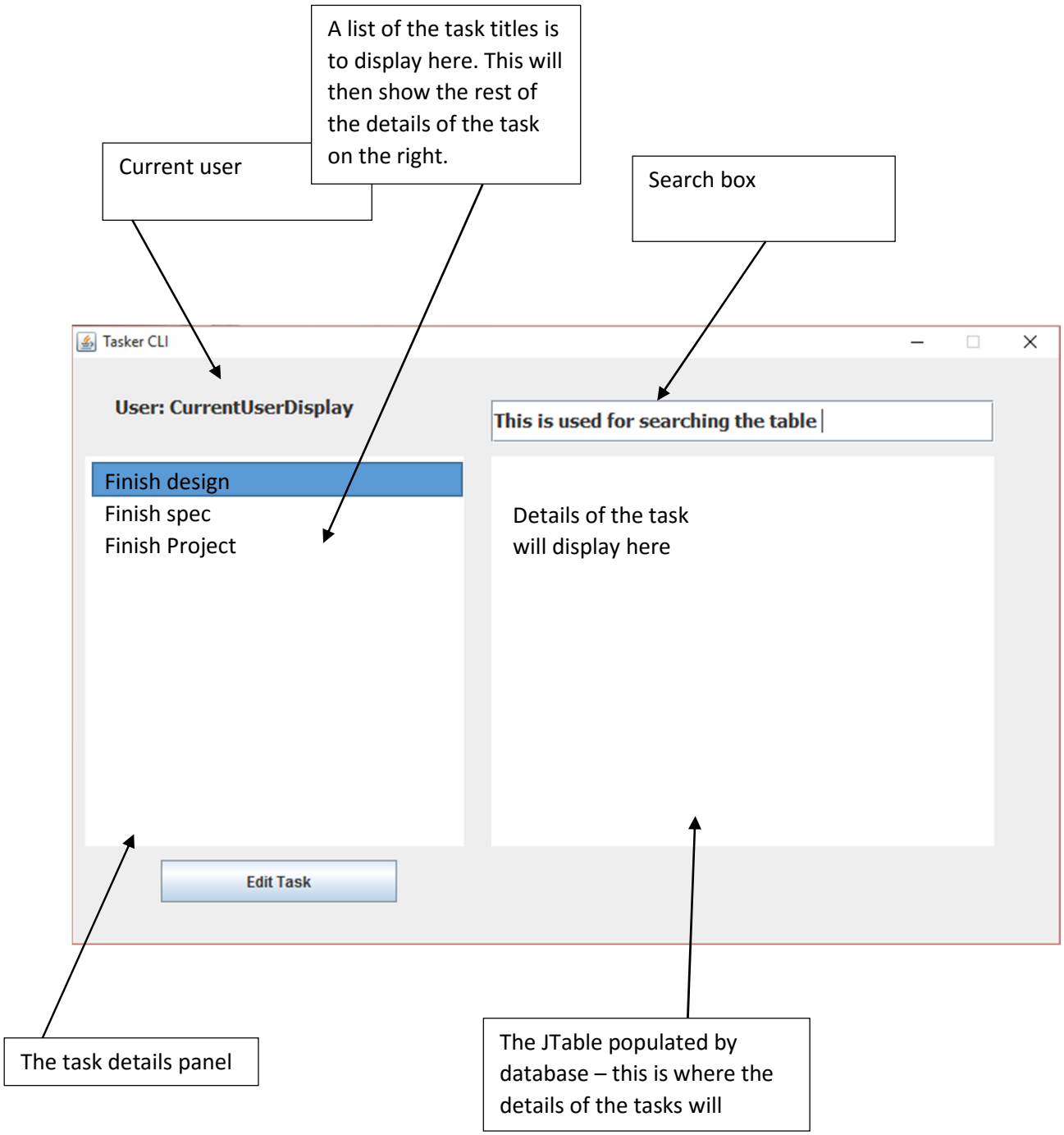
3.2.10a Login page

When the desktop application is first executed, this login windows appears requiring a username and password, which are stored in the database. Alternatively you can login using the offline mode, which uses the files stored on the local machine. Both buttons will redirect you to the user application.



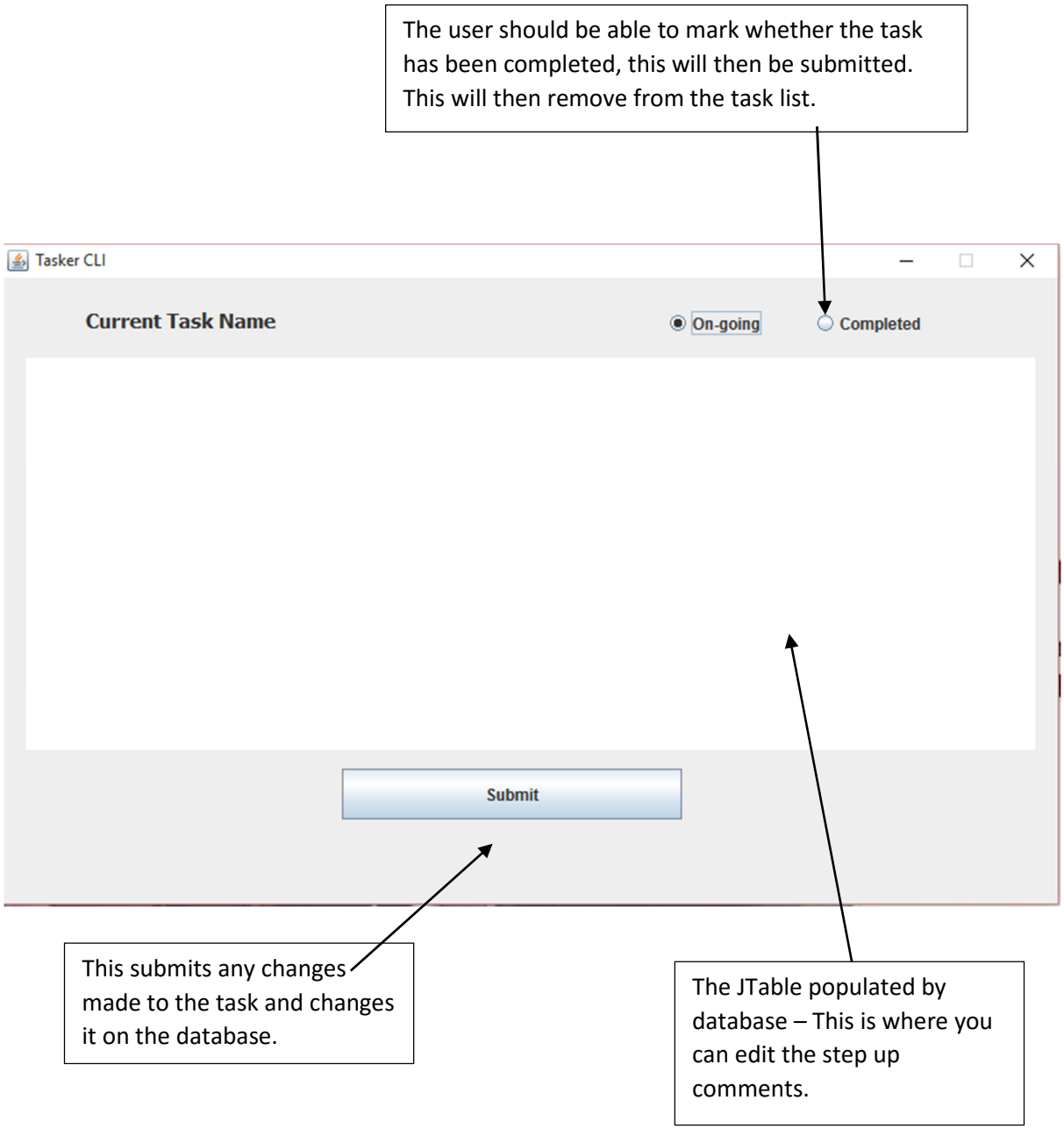
3.2.11 User Application

The user application, is where the main program runs. Here the current user login is displayed, all the current tasks are displayed. Above the table there is search function. The JTable is populated by the database or by local files when running in offline mode. When a task is selected from the table the task details populate the Task details panel. If they wish to edit the task comments they can select the edit task button. The application is to connect to the SQL database and store the tasks allocated for that user, it will only read in the tasks that have not been completed.



3.2.12 Editor

The editor window allows users to edit task details which are populated with data from the database or local files. When you press submit your changes will synchronise with the server, however if you're in offline mode the changes will be saved to local storage and sync to the database as soon possible. You can also set the completion of the task, using the radio buttons. The current selection is retrieved from the database. As the tasks are stored in memory, when a user clicks on edit task, only the comments and status of the task can be edited. When these are edited and submitted it will query with the database and save the new data.



4.0 Significant Classes and Component Description

4.1 MainFrame.java

File Name: MainFrame.java

This class is the main controller of the program, it starts and initialises the program by calling **TaskerLogin.java.**

4.2 Load.java

File Name: Load.java

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.

Functional requirements ref: FR8a FR9

Dependency: DatabaseConnect

4.3 Task.java

File Name: Task.java

Creates an object that is defined by the task properties.

4.4 DatabaseConnect.java

File Name: DatabaseConnect.java

This class is used to connect to our database. 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 accordly. It will then upload any changes that are done locally, and then update the matching tasks on the server with that information.

Functional requirements ref: FR9, FR11

Functional requirements ref: Task.java

4.5 TaskerLogin.java

File Name: TaskerLogin.java

This class creates the graphical user interface for the login part of the application. 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 given the option to use offline mode which uses the local storage while checking for database connection.

Functional requirements ref: FR8

Dependency: MainFrame.java

4.6 TaskerPage.java

File Name: TaskerPage.java

This class displays all the information of all tasks. The information is displayed in a table and when a task is selected it's 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

4.7 TaskerEditor.java

File Name: TaskerEditor.java

Creates the editor window for tasks. 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.

Functional requirements ref: FR10

Dependency: TaskerPage.java

4a.1 PHP files

4a.1.1 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 the file: taskerMAN/index.php

4a.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 will be no php functions used on this page.

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

4a.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 will re-direct 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.

4a.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

4a.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

4a.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

4a.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)

4a.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

4a.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

4a.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

4a.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

4a.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

4a.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)

4a.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

4a.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)

4a.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

4a.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

4a.1.18 Error.php

File Name: e-rror.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.

4a.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

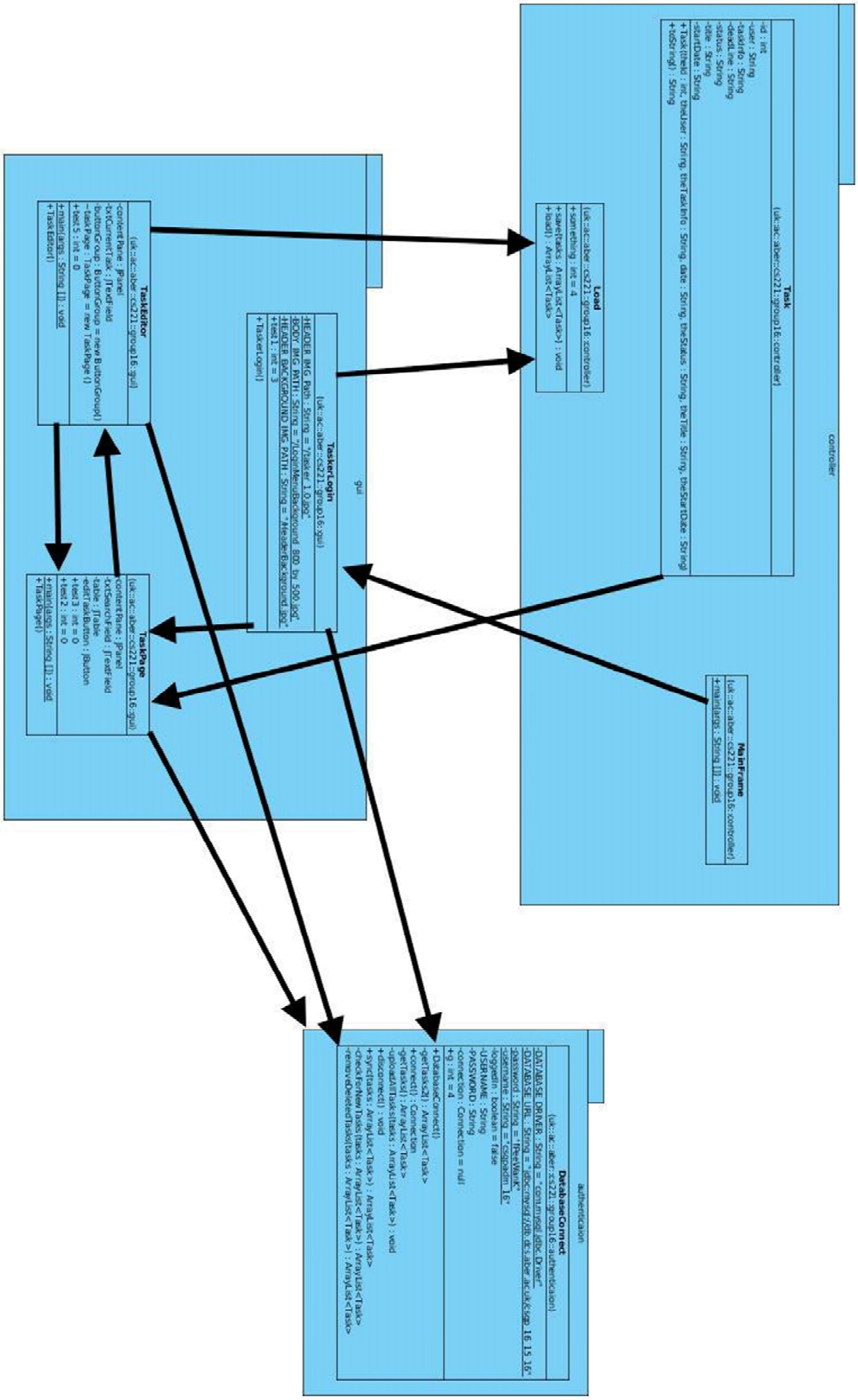
4a.1.8 RemoveMember.php

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


controller
Task
(uk::ac::aber::cs221::group16::controller)
-id : int
-user : String
-taskInfo : String
-deadline : String
-status : String
-title : String
-startDate : String
+Task(theId : int, theUser : String, theTaskInfo : String, date : String, theStatus : String, theTitle : String, theStartDate : String)
+toString() : String
Load
(uk::ac::aber::cs221::group16::controller)
+something : int = 4
+saveTasks(ArrayList<Task>) : void
+load() : ArrayList<Task>
MainFrame
(uk::ac::aber::cs221::group16::controller)
+main(args : String []) : void
gui
TaskEditor
(uk::ac::aber::cs221::group16::gui)
-contentPane : JPanel
-txtCurrentTask : JTextField
-buttonGroup : ButtonGroup = new ButtonGroup()
-taskPage : TaskPage = new TaskPage ()
+test5 : int = 0
+main(args : String []) : void
+TaskEditor()
TaskerLogin
(uk::ac::aber::cs221::group16::gui)
+HEADER_IMG_Path : String = "/tasker_1.0.png"
-BODY_IMG_PATH : String = "/LoginMenuBackground_800_by_500.png"
+HEADER_BACKGROUND_IMG_PATH : String = "/HeaderBackground.png"
+test1 : int = 3
+TaskerLogin()
TaskPage
(uk::ac::aber::cs221::group16::gui)
-contentPane : JPanel
-txtSearchField : JTextField
-table : JTable
-editTaskButton : JButton
+test3 : int = 0
+test2 : int = 0
+main(args : String []) : void
+TaskPage()
authentication
DatabaseConnect
(uk::ac::aber::cs221::group16::authentication)
-DATABASE_DRIVER : String = "com.mysql.jdbc.Driver"
-DATABASE_URL : String = "jdbc:mysql://db.dcs.aber.ac.uk/cs221_16_15_16"
-password : String = "theeWenK"
-username : String = "cs221admin_16"
-loggedIn : boolean = false
-USERNAME : String
-PASSWORD : String
-connection : Connection = null
+g : int = 4
+DatabaseConnect()
-getTasks(Id) : ArrayList<Task>
+connect() : Connection
-getTasks() : ArrayList<Task>
-uploadAllTasks(tasks : ArrayList<Task>) : void
+disconnect() : void
+sync(tasks : ArrayList<Task>) : ArrayList<Task>
-checkForNewTasks(tasks : ArrayList<Task>) : ArrayList<Task>
-removeDeletedTasks(tasks : ArrayList<Task>) : ArrayList<Task>

5.0 Detailed Design

5.1 Schema Design

This is the design for the databases and will allow for JDBC and PHP library calls to be made to retrieve appropriate data.

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.

Within our Group Project we decided we needed to have two database tables, one for the members and one for the tasks. We chose to have two, one for the login which will be our members table, we will use the email address field as the username and the password field as the password to login. The other fields will be used to hold basic information and the Primary Key for a unique identifier. We decided to have all of the data types set to text apart from the 'id' which is set to int(11) for simplicity.

Our second and final database table is Tasks, this table will be used to store data about the set tasks which will be allocated to members from the members table. Again, all data types are set to varchar with various sizes for simplicity apart from the TaskID which is used to uniquely identify the Task given. We were going to have some data types set to Date, but decided not to and are going to use javascript and various methods within java to validate the input so there is no room for error.

5.3 Format of Data Transmitted

SQL Database

An SQL server set up on "db.dcs" receives and sends data to and from the applications. Data is being passed from one system/application, this includes data being sent from the website through PHP and the java application. Both of these are passing information to and from the SQL server. The website and Java application are sending SQL queries to the server, this is how the system requests or adds new data to and from the server an example of this is seen below:

```
SELECT
        *
FROM
        members
WHERE
        id='5'
```

Data is also being sent back from the SQL server to the applications of the system. This is being sent as a string and the system interprets this.

Local Storage for offline use

All the appropriate tasks are stored as objects in an arraylist. This object (tasks) is to be stored as:

- Id – for the identification.
- User – name of the account.
- Taskinfo – information of the task.
- StartDate – when the task was given.
- Deadline – when the task needs to be completed.
- Status – whether or not the task is completed.
- Title – the title of the task.

This arraylist is then used to save the data into a file on local storage which will send the updated data next time it is online. The format that the data should be saved is:

- 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.

This should then be sent to the database when a connection is made.

5.4 Difficult Decisions

One of the difficult decisions was figuring out the layout of the web application, while trying to fulfil all the functional requirements of the requirements specification. Another difficult decision we came across was working what table we needed for the database and what the structure of these tables should include, for example what columns we needed and what type and constraints to use.

References

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

Change History

Version	CCF No.	Date	Changes Made To Document	Changed By
1.1	N/A	2015-10-29	Format changed to suit design specification. Changed release version.	Robert Mouncer – rdm10
1.2	N/A	2015-10-29	Changed to release.	Robert Mouncer – rdm10
1.3	N/A	2015-11-25	Changed to draft after Nigel Hardy's feedback	Robert Mouncer –rdm10
1.4	N/A	2015-11-25	Ready For Review after updated for new specification	Robert Mouncer –rdm10
1.5	N/A	2015-11-25	Ready For Release after changes made.	Robert Mouncer –rdm10
1.6	N/A	2016-01-25	Changes made to suit feedback from Nigel Hardy	Robert Mouncer –rdm10