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.

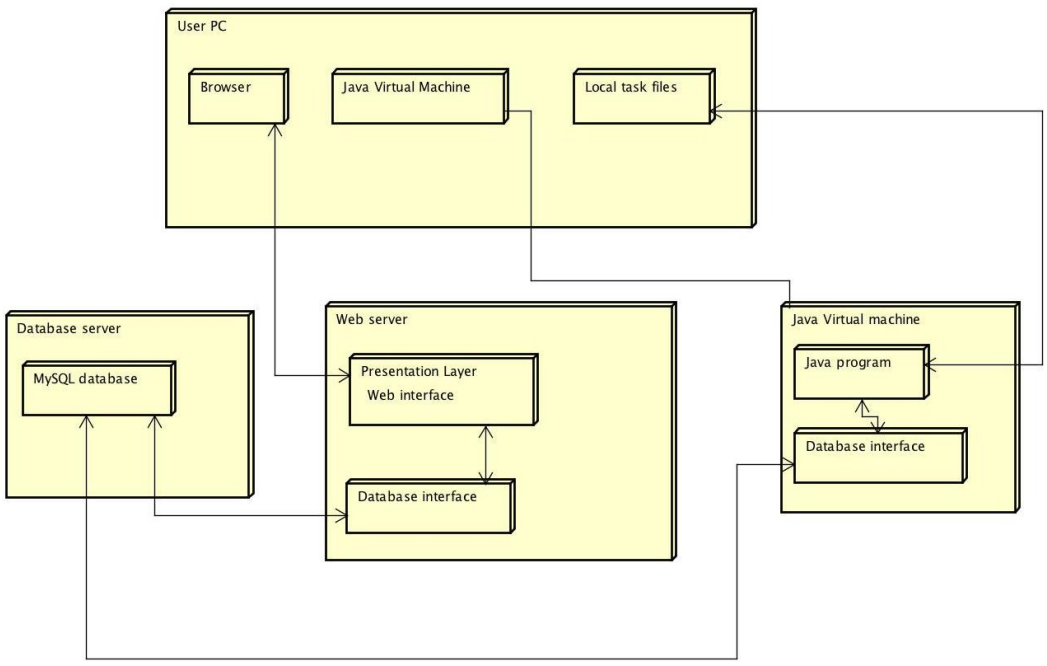
1.3 Objectives

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.



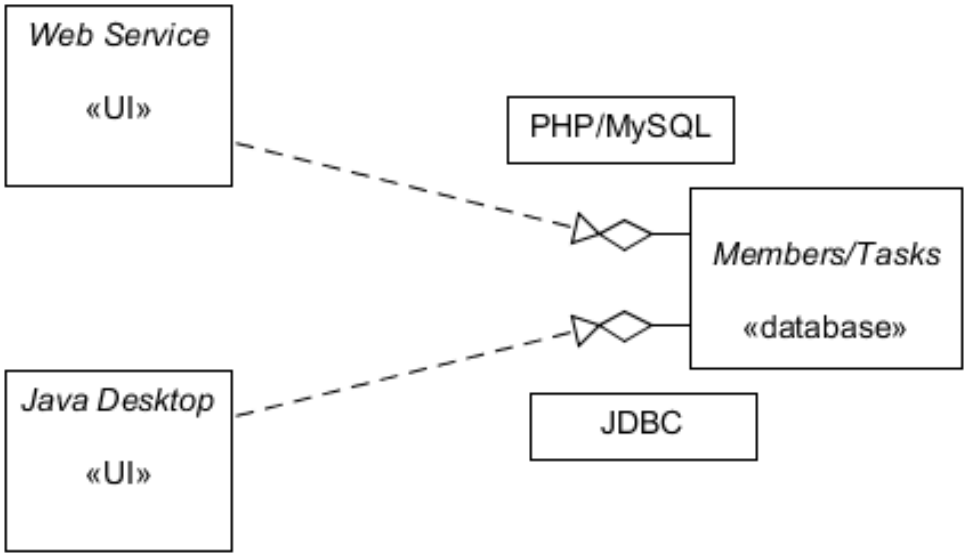
2.2 Applications Interactions

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.

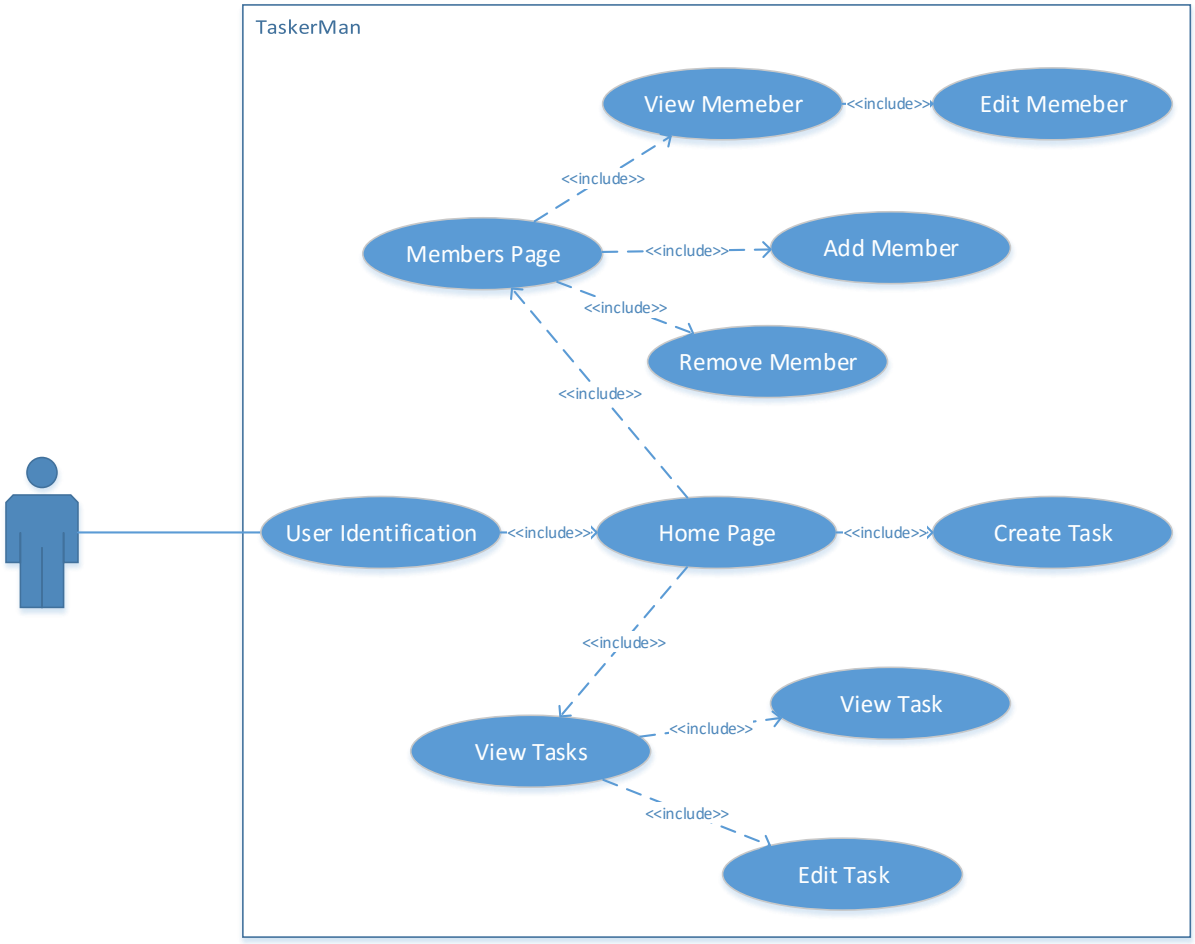
Below is a component diagram illustrating everything that is mentioned above.



3.0 Interaction Design

3.1 Use Case Diagrams

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



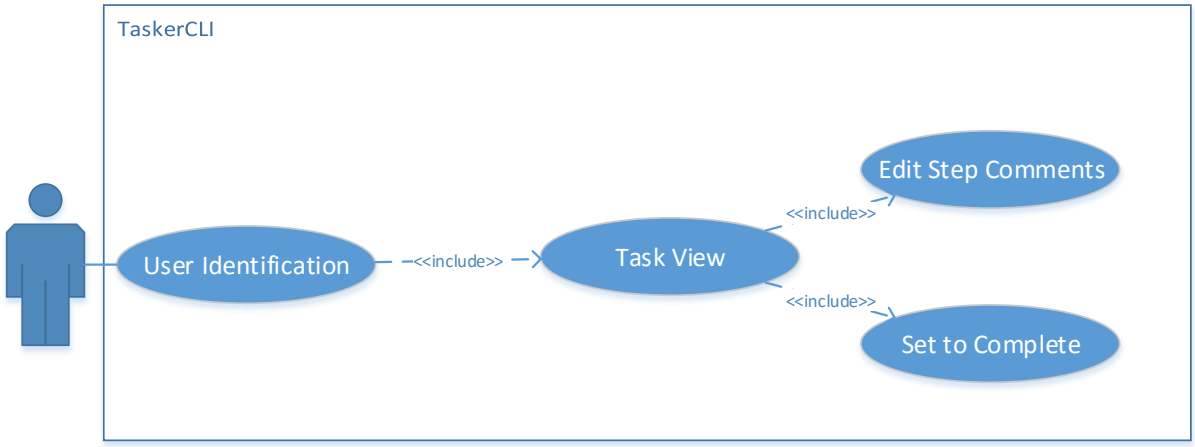
The name of the system (TaskerMAN), is displayed on the top left within the System boundary.The Actor in this case is the Manager of the project, which can be seen to the left of the use case, just outside the system boundary.

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’.

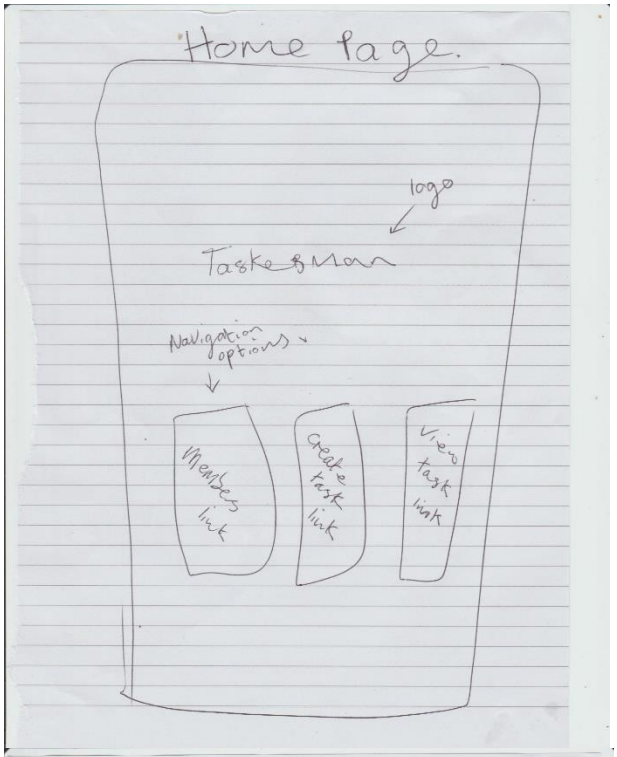
Below is the use case diagram for the user application Tasker CLI.

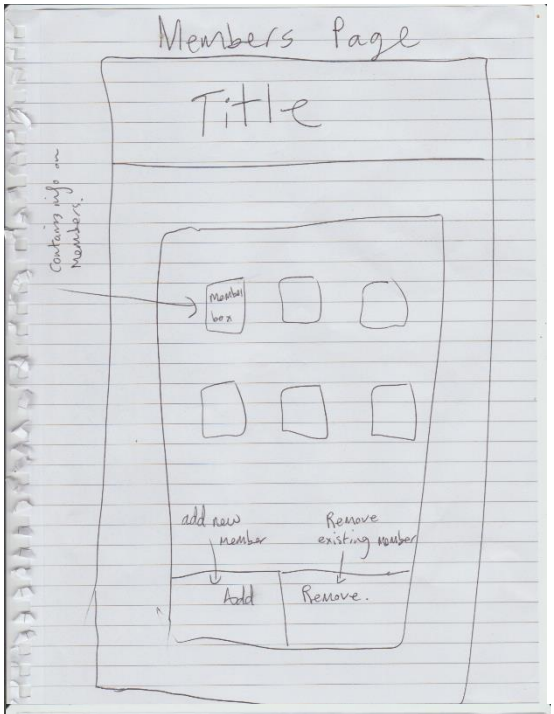


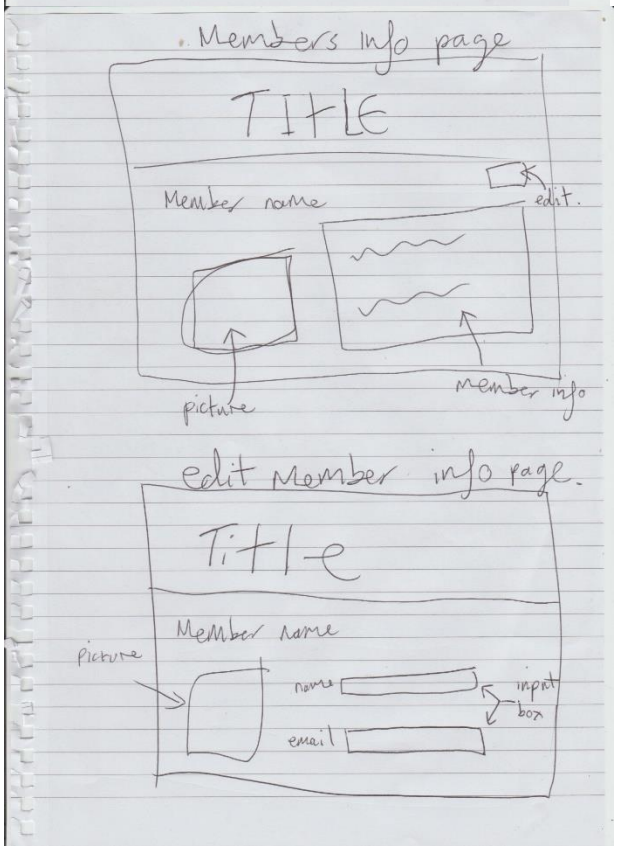
The associations are used in the use case diagram are used to display how the use interacts with the application. You can view all the functionality of the program. So we can see that the application will provides: user identification, the ability to view tasks, to edit the comment about the tasks and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

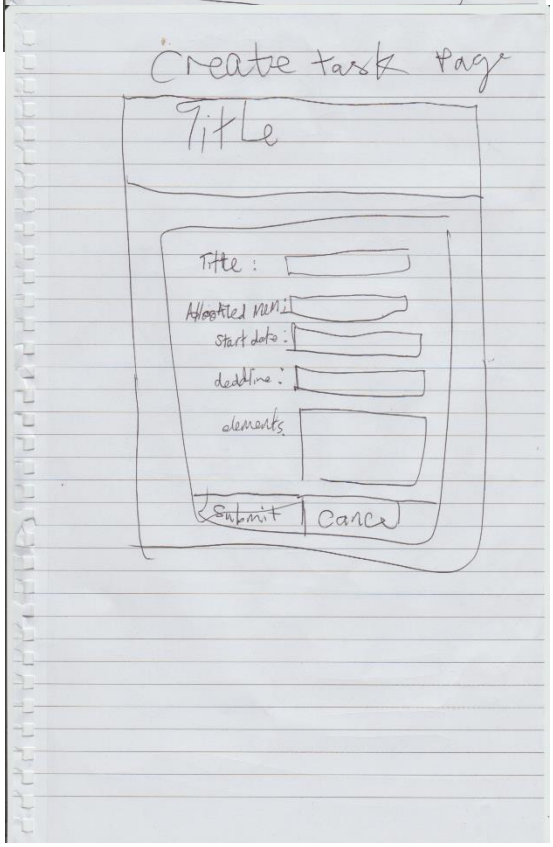
3.2 User Interface Design

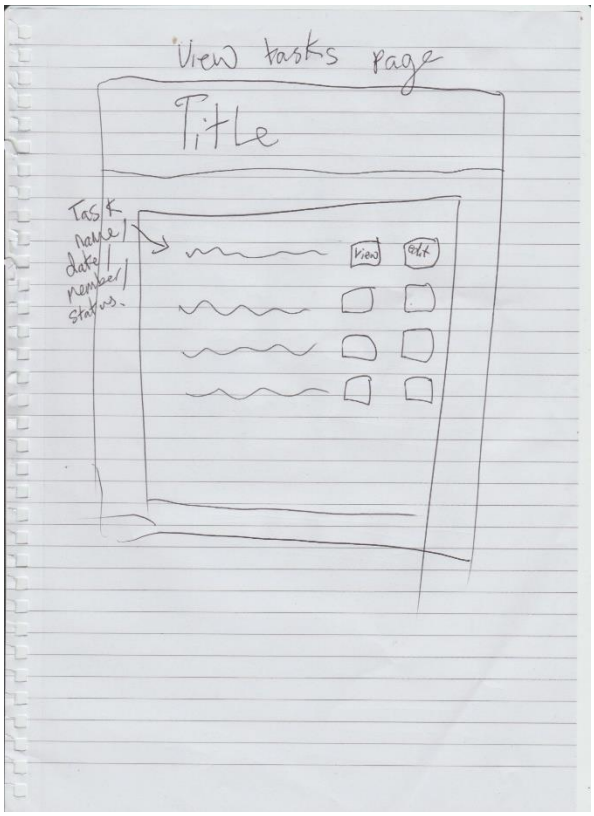
3.2.1 Rough TaskerMAN Designs

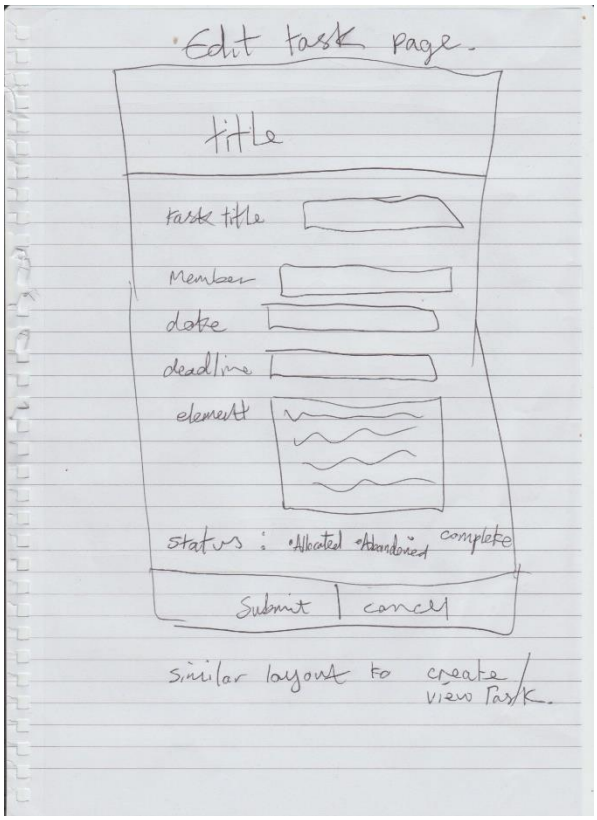












These designs are a rough representation of the layout of TaskerMAN. These will be used to help create the design of the website.

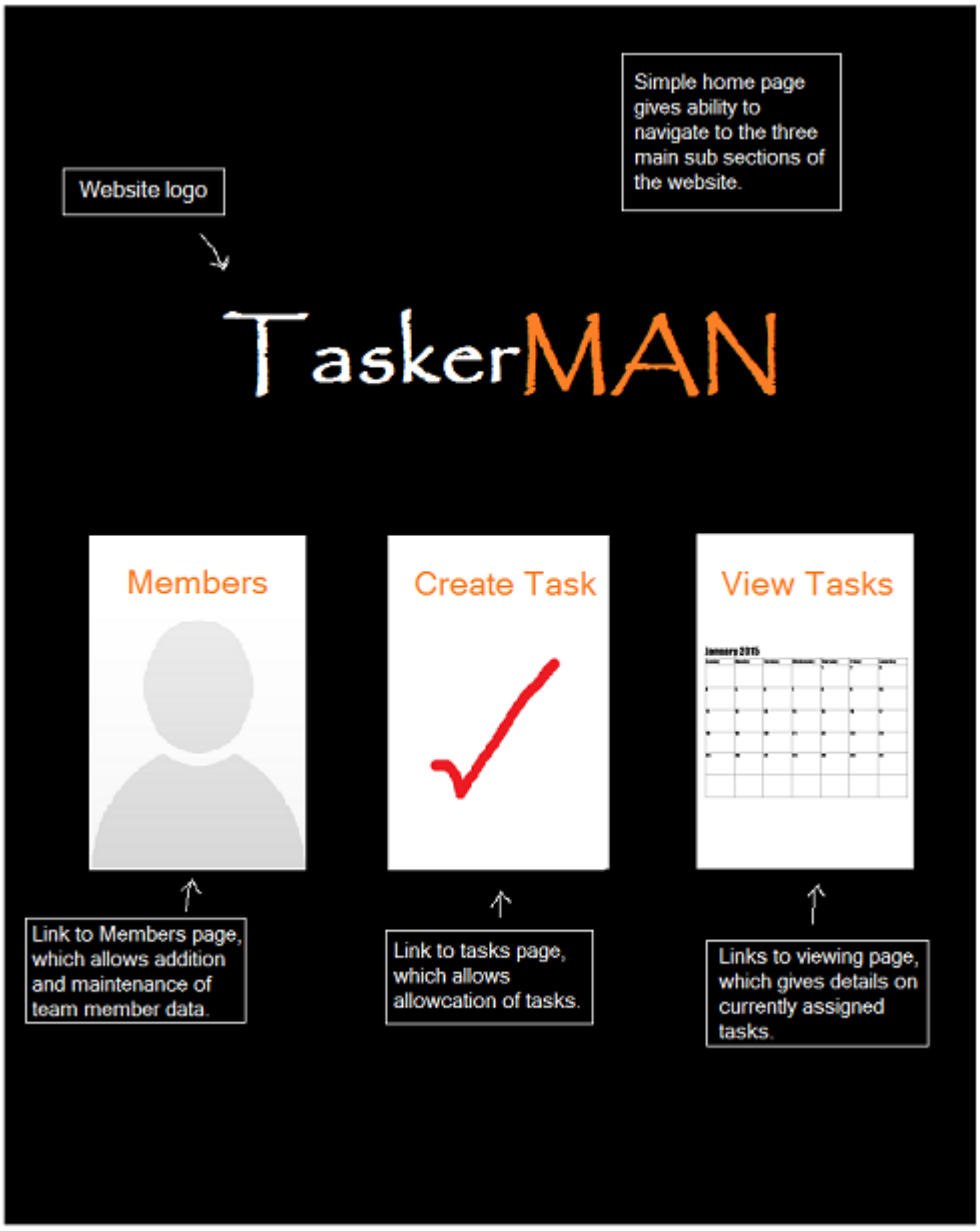
3.2.1a Template

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

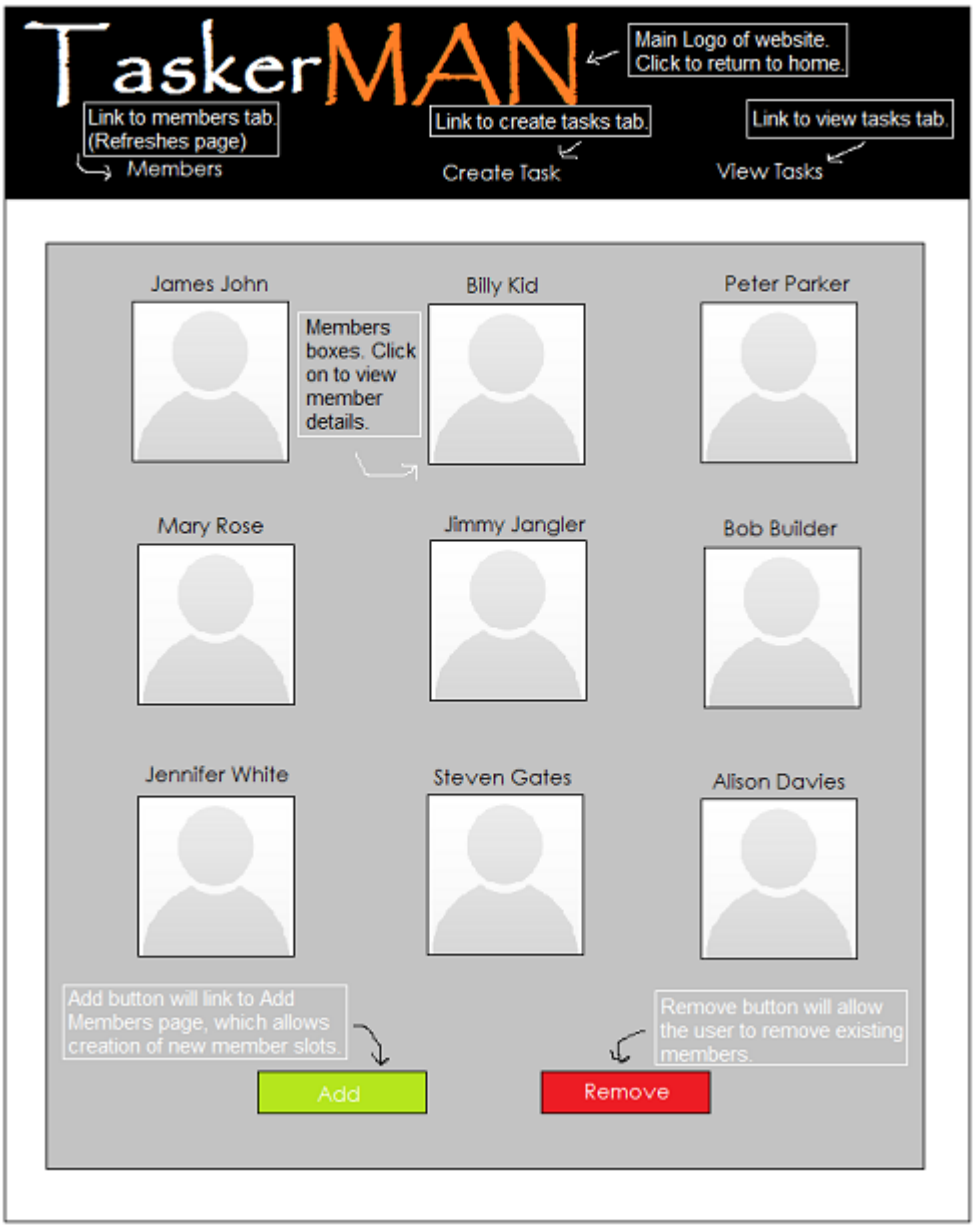


3.2.3 Members Page

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

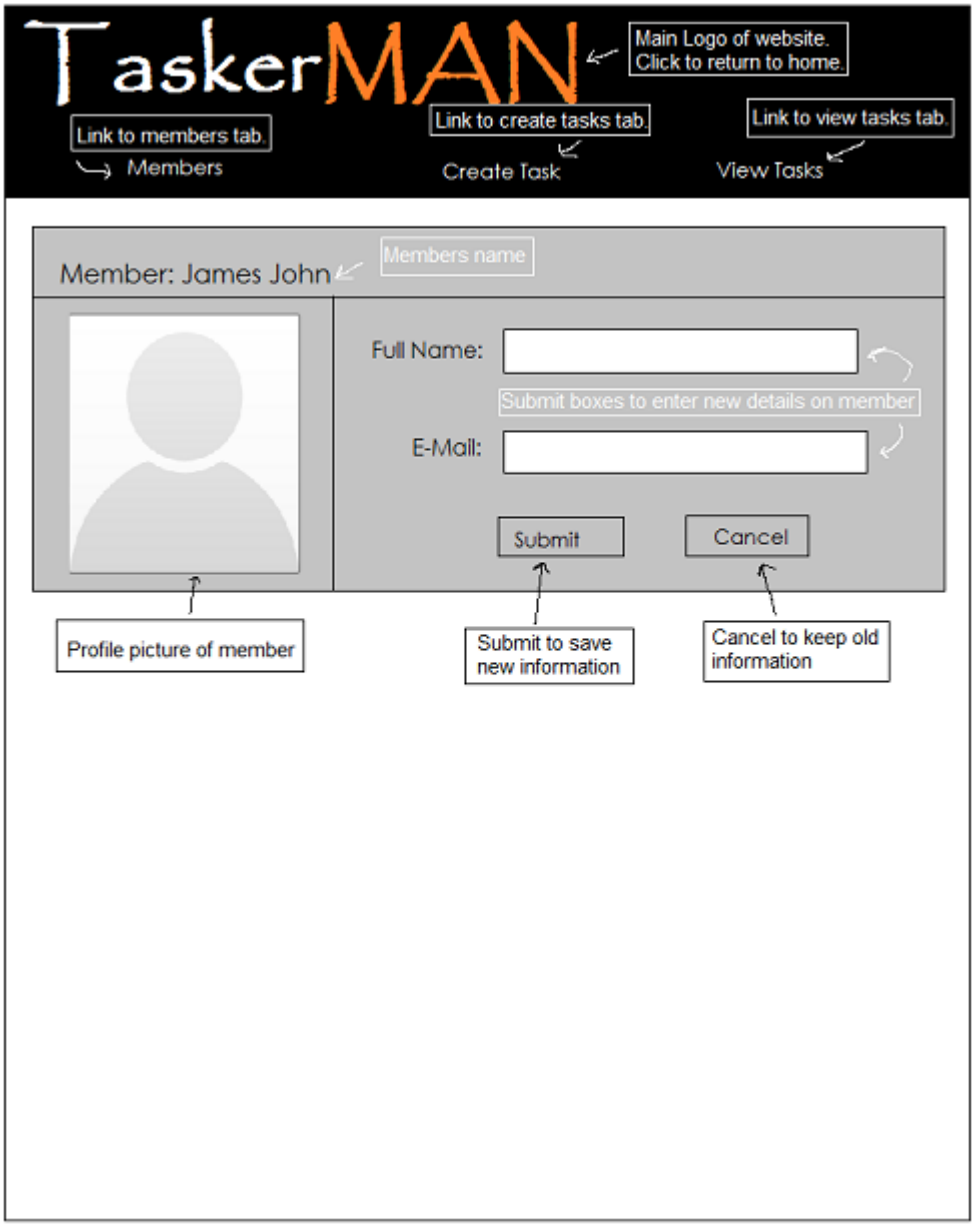
The information page contains the required details of the member. The edit button on the top right of the users information is a link to the 'Edit Members Information Page', which gives the user the ability to the existing information on the member.
The URL for this page will be:

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

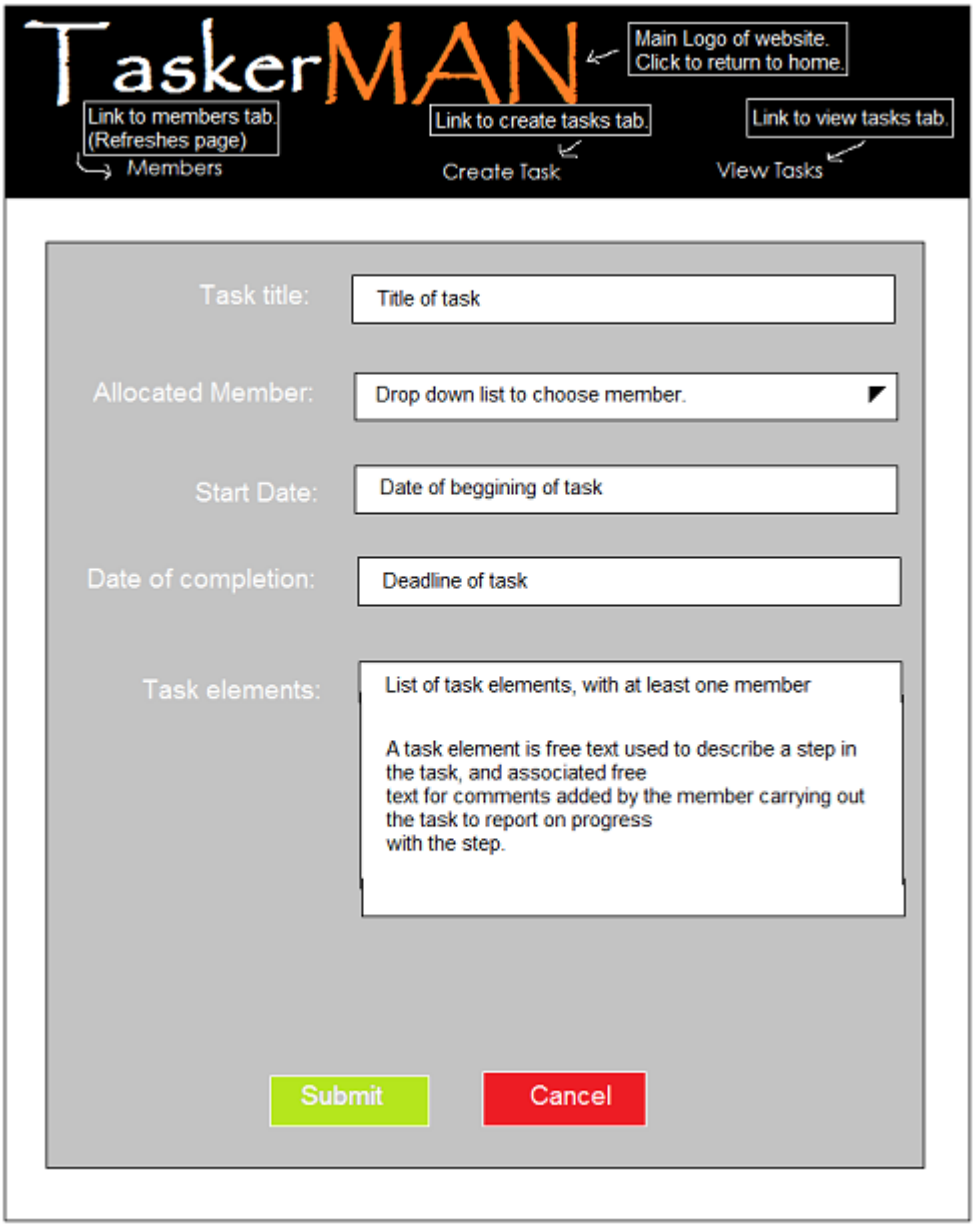


3.2.6 Create Task Page

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

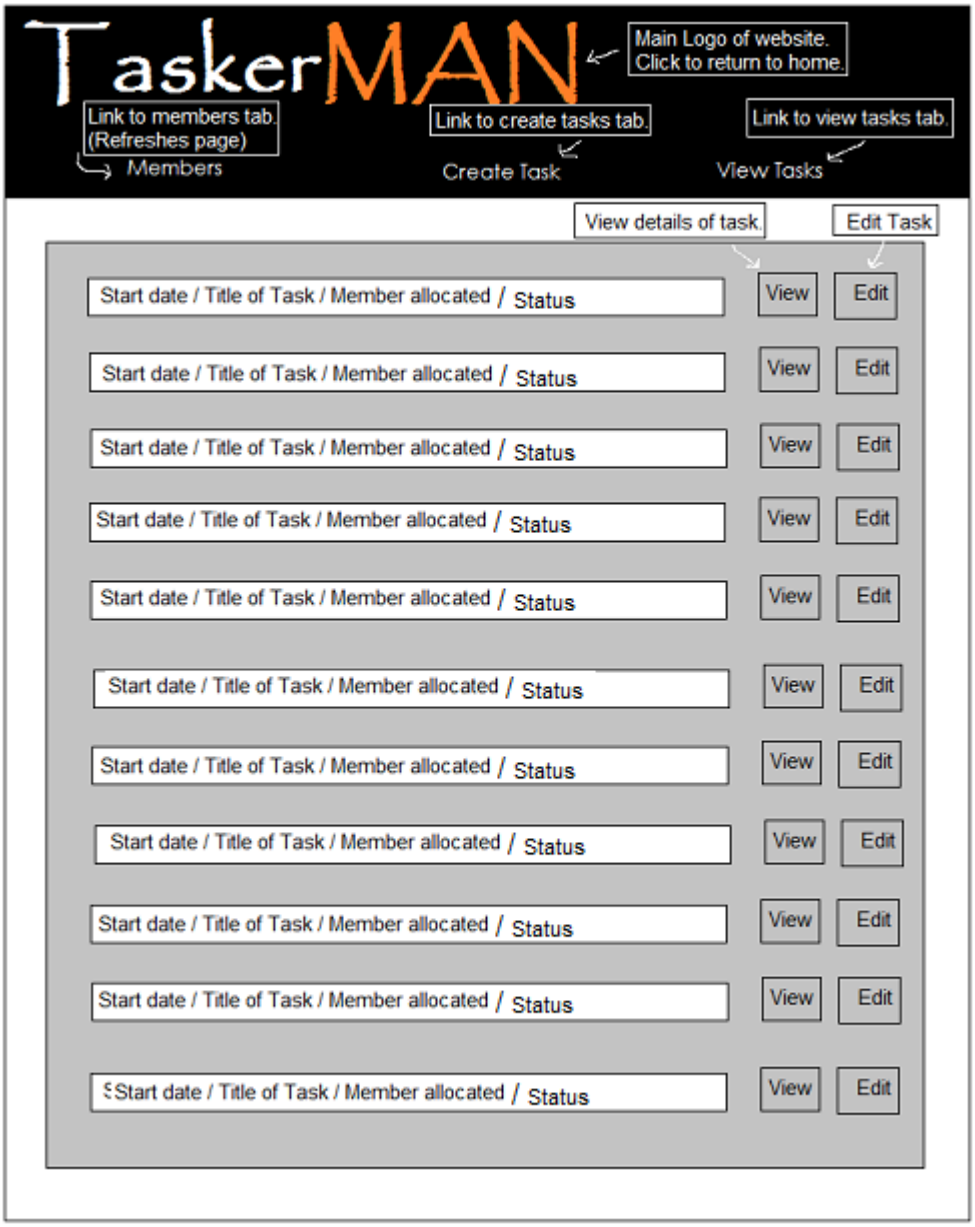


3.2.7 View Tasks Page

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:

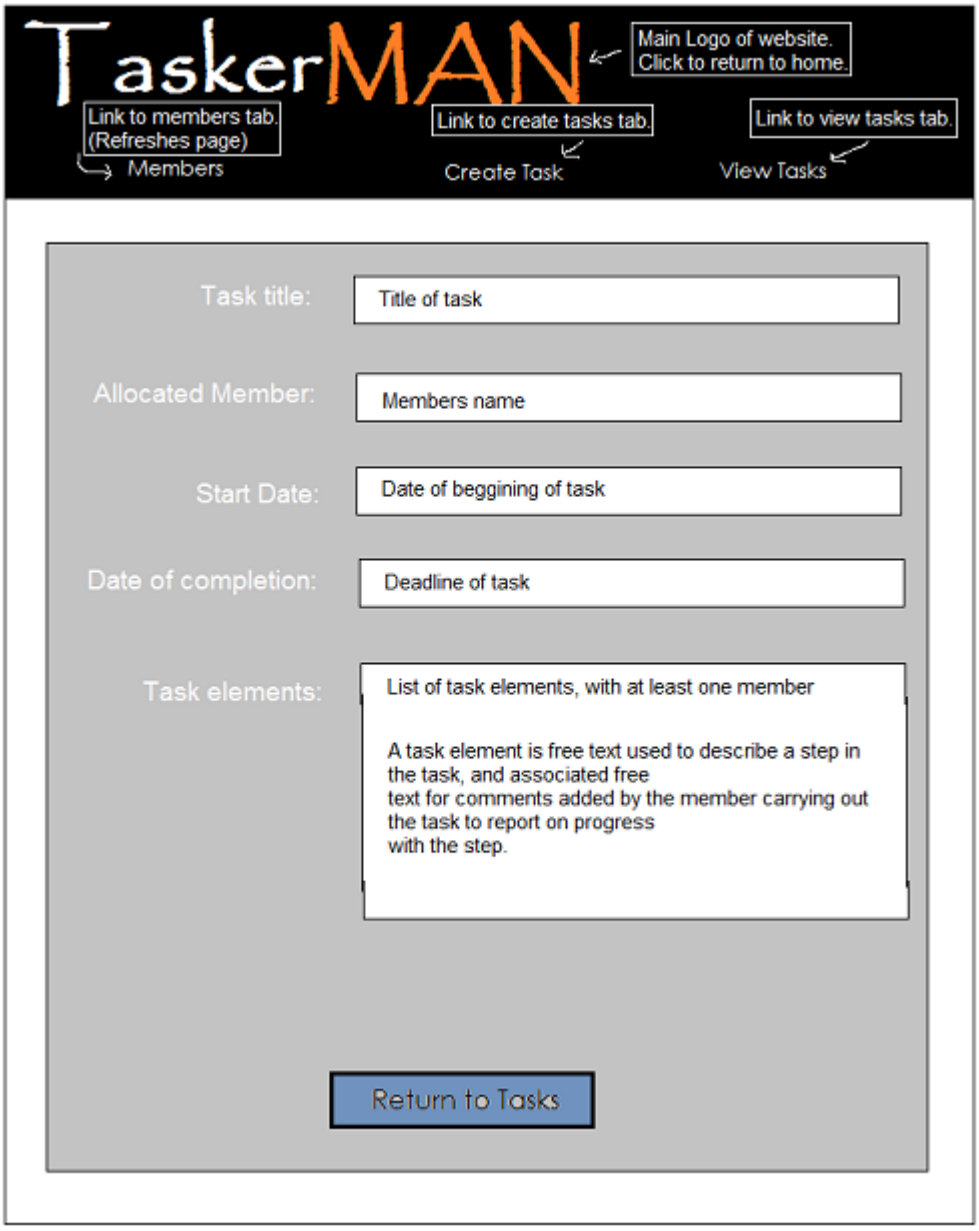
http://users.aber.ac.uk/abc123/ taskerMAN/viewtasks.php



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:

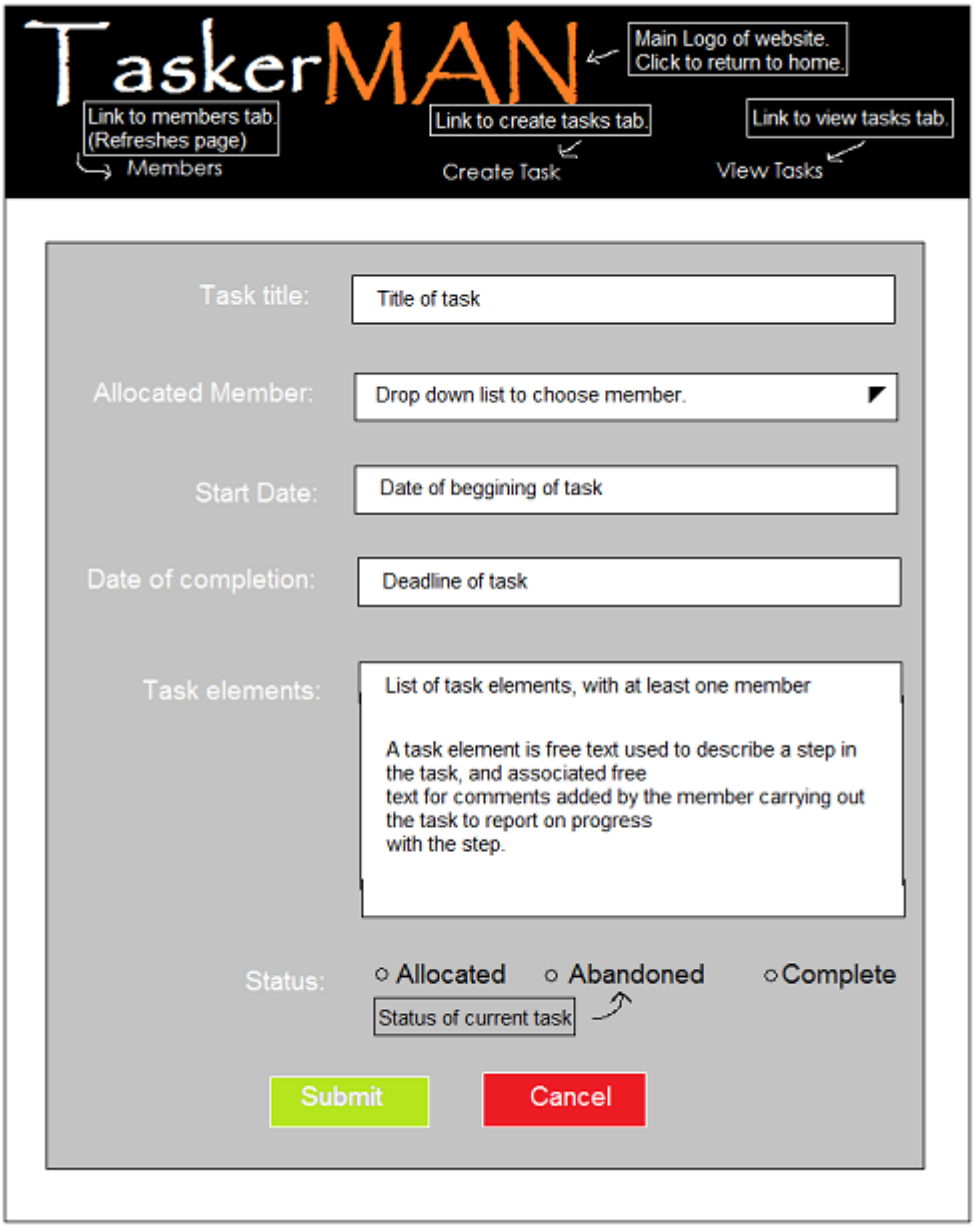
http://users.aber.ac.uk/djt/taskerMAN/viewTask.php?Taskid=52



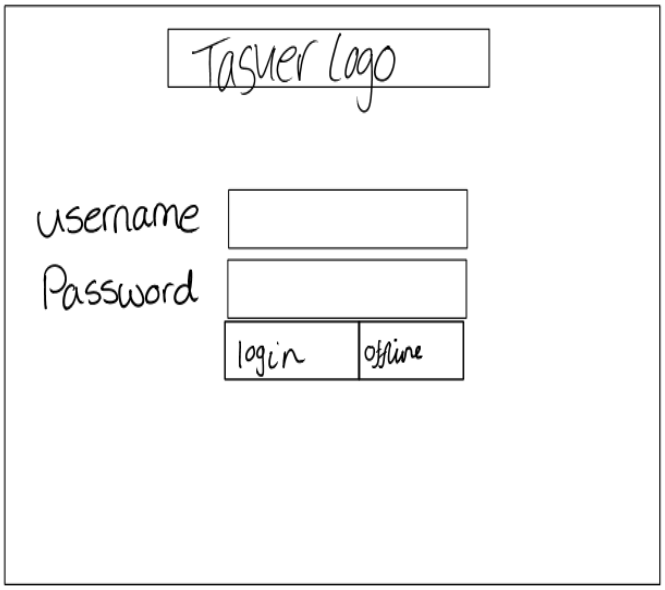
3.2.9 Edit Task Page

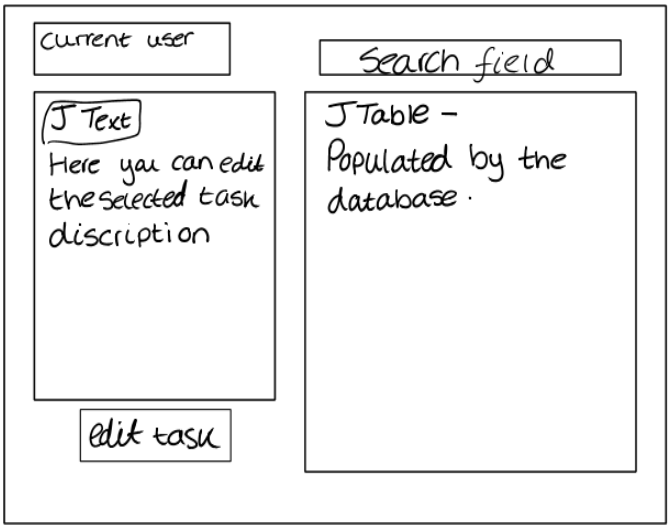
The edit task page will give all the same options as was had when creating the task originally, with the addition allowing the user to edit the tasks current status. From here the user will also be able to change the member to whom the task is allocated. The 'Submit' button will save any changes made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

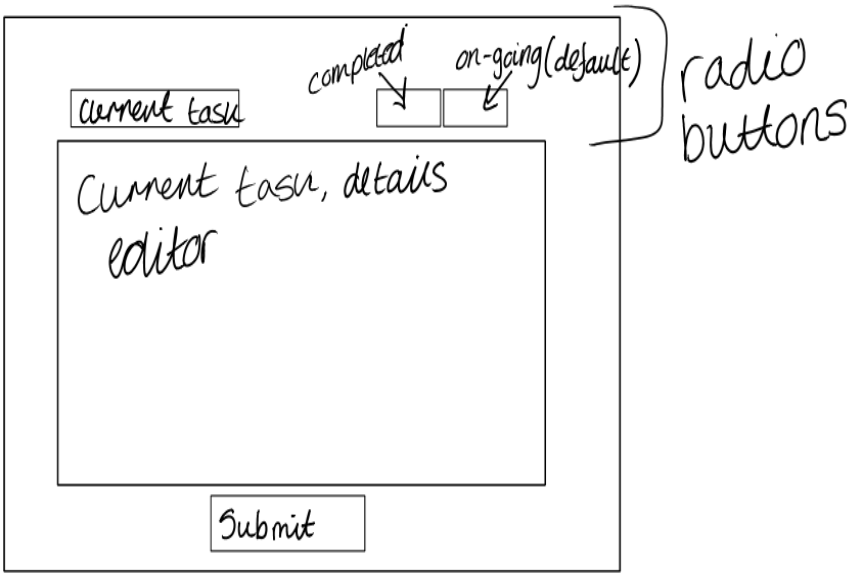
http://users.aber.ac.uk/djt/taskerMAN/editTask.php?TaskID=3



3.2.10 Rough TaskerCLI Designs



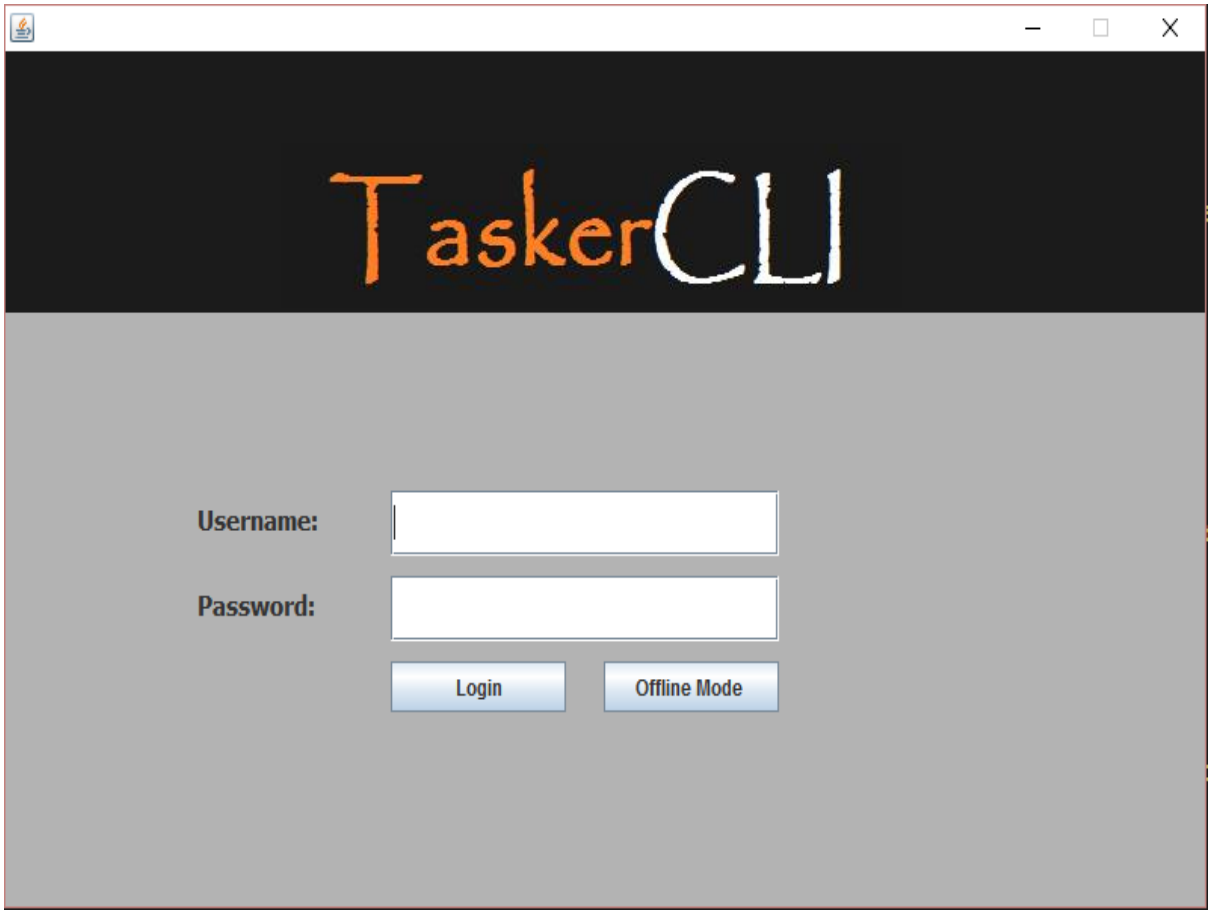




Below are the final designs for the user interface on the user application Tasker CLI

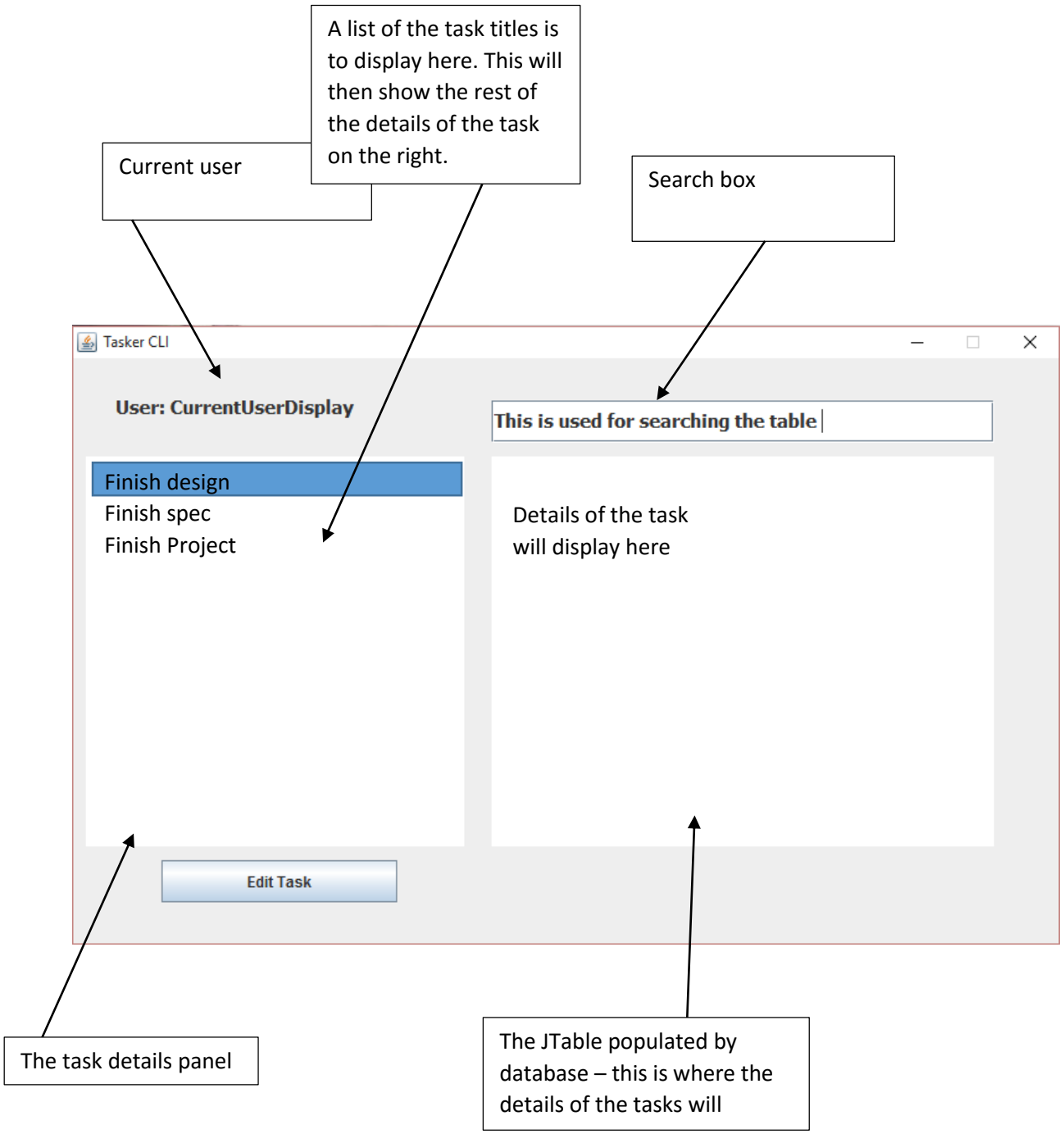
3.2.10a Login page

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.



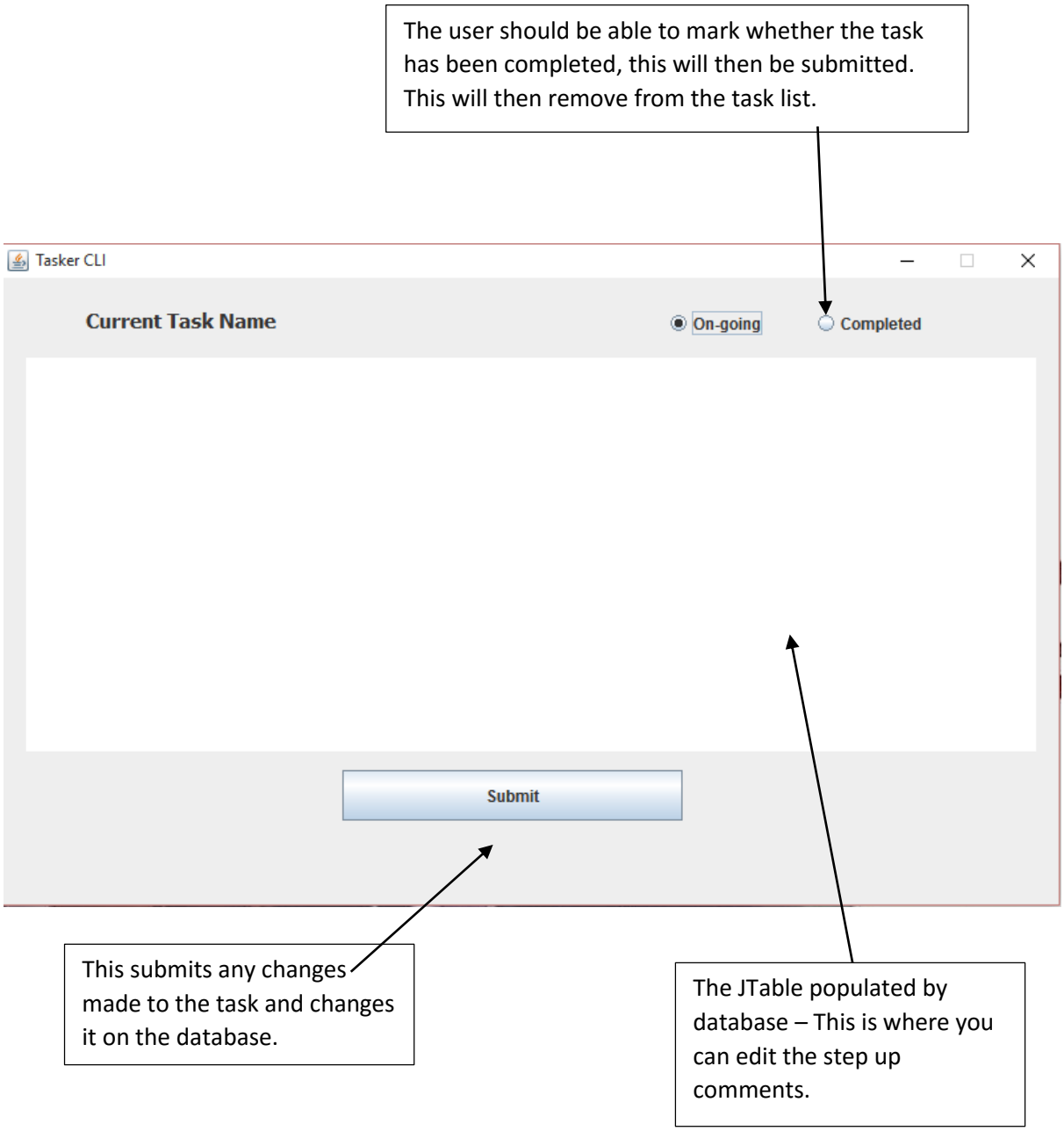
3.2.11 User Application

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

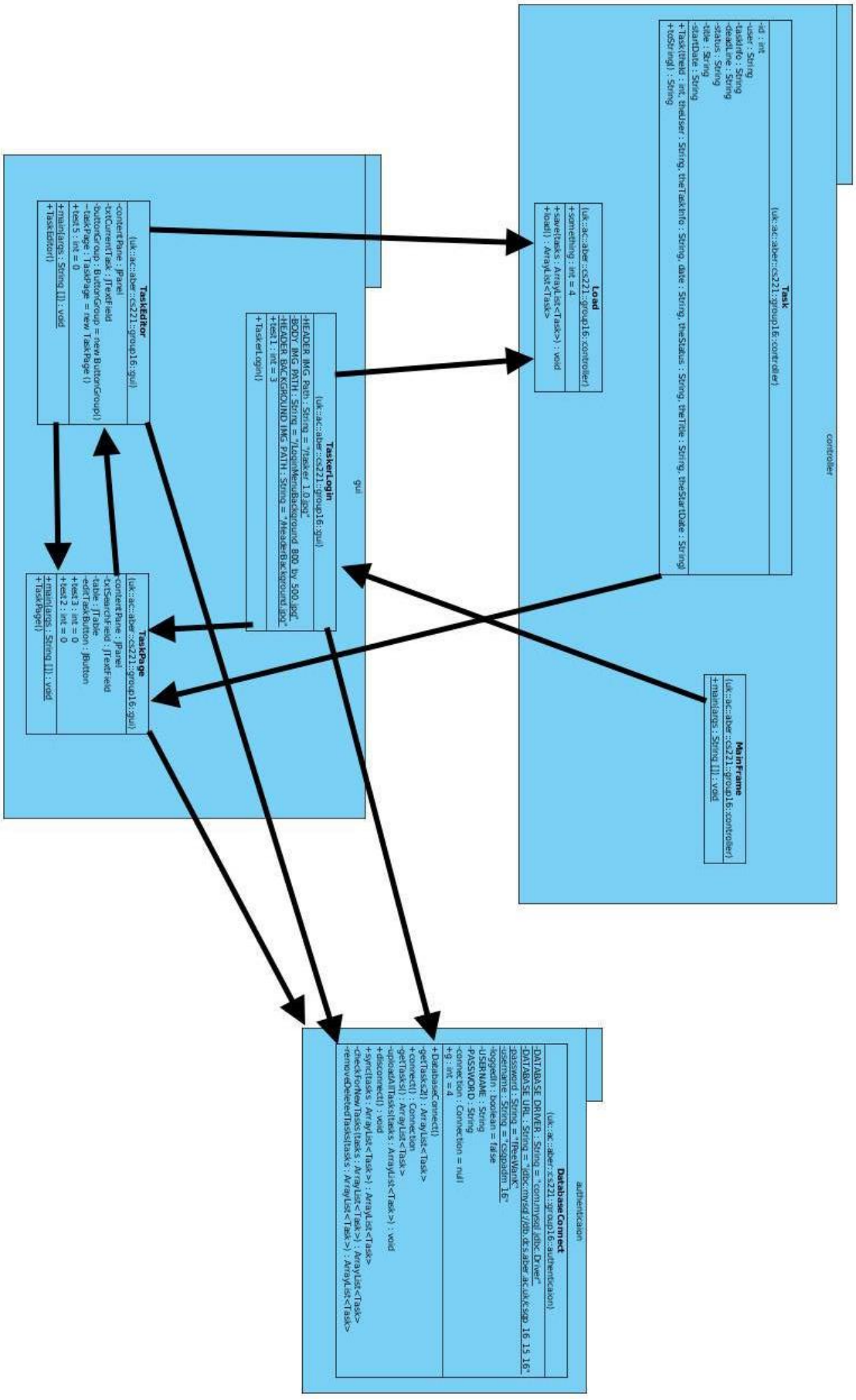
File Name: TaskerEditor.java

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


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
+save(tasks : 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.jpg"
-BODY_IMG_PATH : String = "/taskerMenuBackground_800_by_500.jpg"
-HEADER_BACKGROUND_IMG_PATH : String = "/HeaderBackground.jpg"
+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.cs.aber.ac.uk:cspp_16_15_16"
-password : String = "PeeWarnK"
-username : String = "cspadm_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