

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: 1.5

Status: Released

Department of Computer
Science Aberystwyth
University Aberystwyth
Ceredigion SY23 3DB
Copyright © Aberystwyth
2015

Contents

1.0 Introduction	3
1.1 Purpose of this document	3
1.2 Scope	3
1.3 Objectives	3
2.0 Deployment Description	3
2.1 Applications in the system	3
2.2 Applications Interactions.....	4
3.0 Interaction Design	5
3.1 Use Case Diagrams	5
3.2 User Interface Design	7
3.2.1 Template	9
3.2.2 Home Page	9
3.2.3 Members Page	10
3.2.4 Members Information Page	11
3.2.5 Edit Members Information Page	12
3.2.6 Create Task Page	13
3.2.7 View Tasks Page	14
3.2.8 View Task Page	15
3.2.9 Edit Task Page	16
3.2.10 Login page	18
3.2.11 User Application.....	19
3.2.12 Editor.....	20
4.0 Significant Classes and Component Description.....	21
4.1 MainFrame.java.....	21
4.2 Load.java	21
4.3 Task.java	21
4.4 DatabaseConnect.java.....	21
4.5 TaskerLogin.java	22
4.6 TaskerPage.java	22
4.7 TaskerEditor.java	22
5.0 Detailed Design	24
5.1 Schema Design	24
5.2 Format of Data Transmitted.....	24
5.3 Difficult Decisions	24
References	25
Change History	25

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.

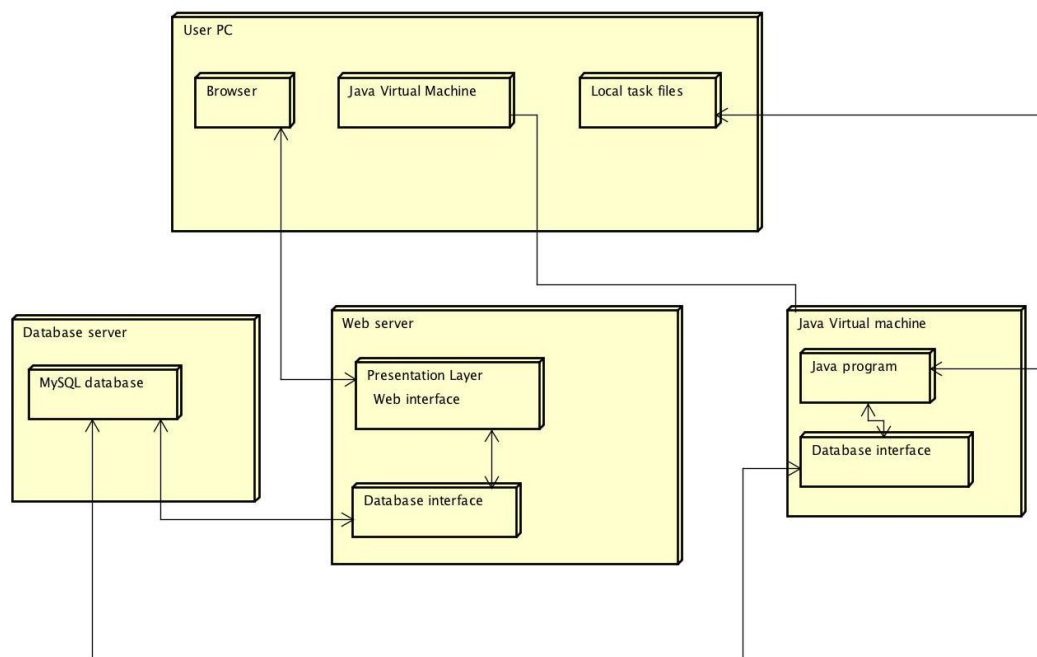
1.3 Objectives

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.



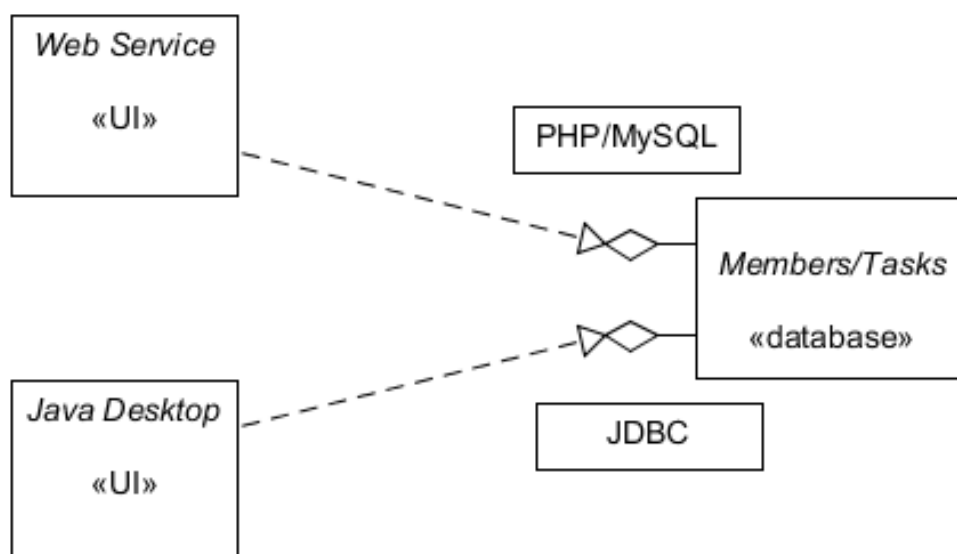
2.2 Applications Interactions

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.

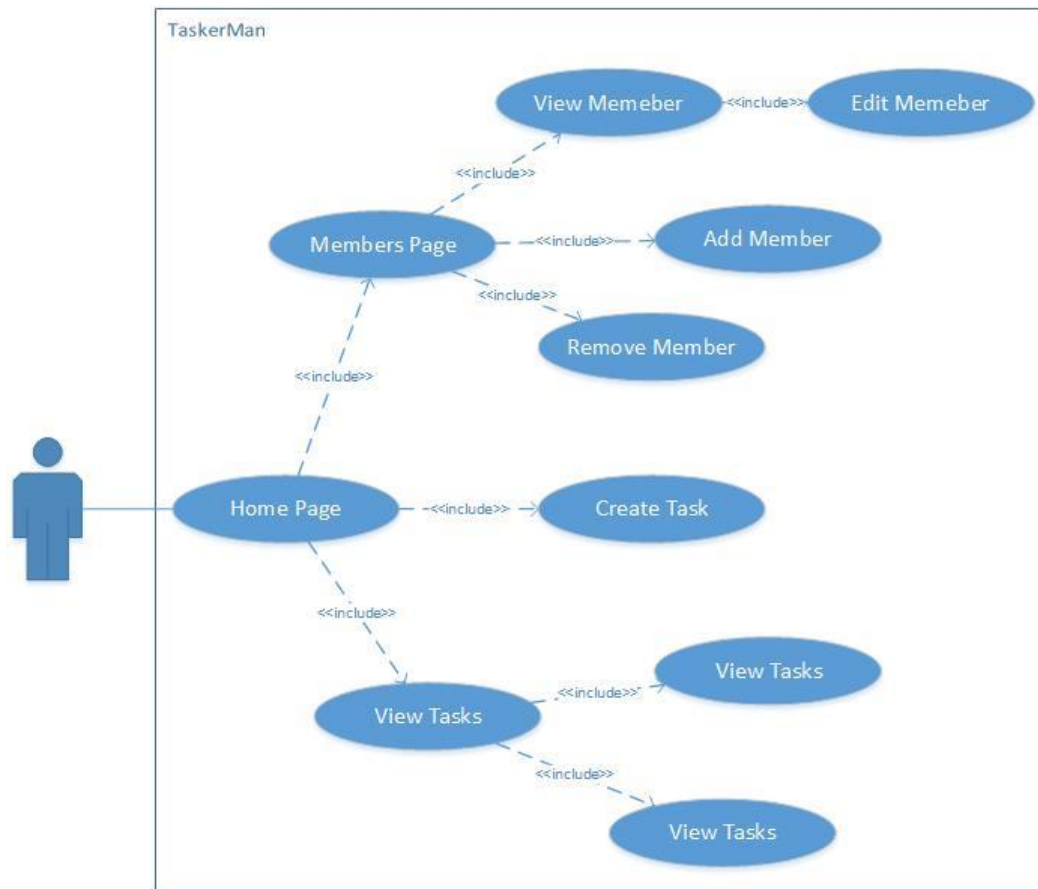
Below is a component diagram illustrating everything that is mentioned above.



3.0 Interaction Design

3.1 Use Case Diagrams

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



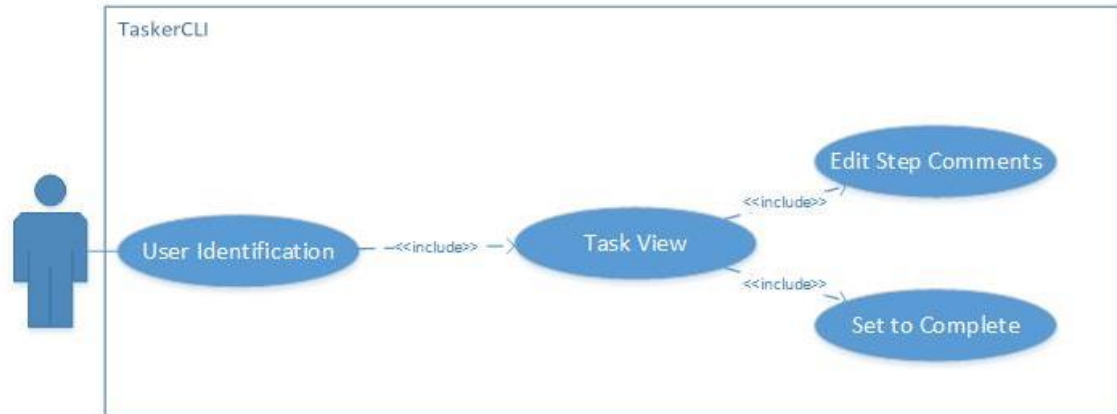
The name of the system (TaskerMAN), is displayed on the top left within the System boundary. The Actor in this case is the Manager of the project, which can be seen to the left of the use case, just outside the system boundary.

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'.

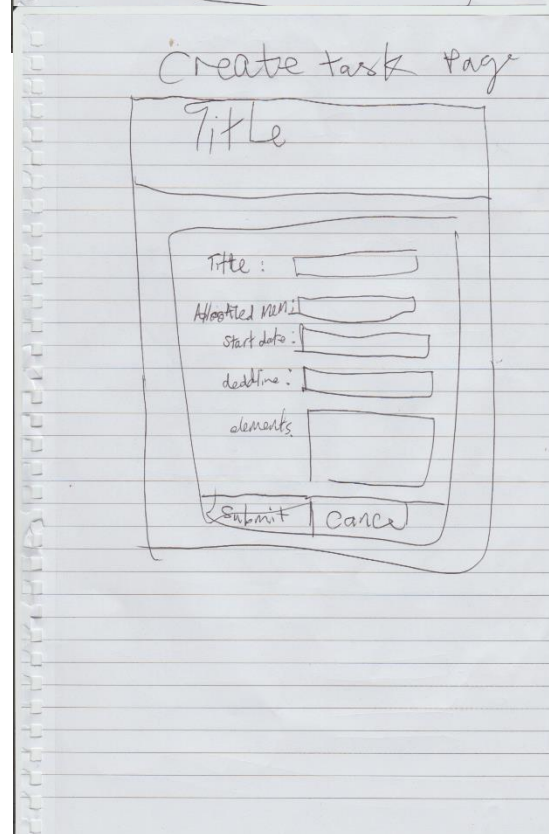
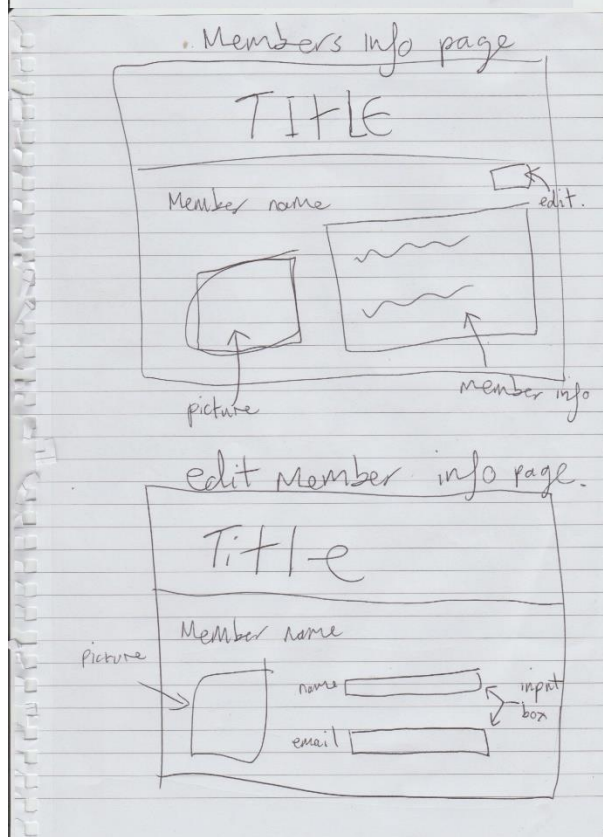
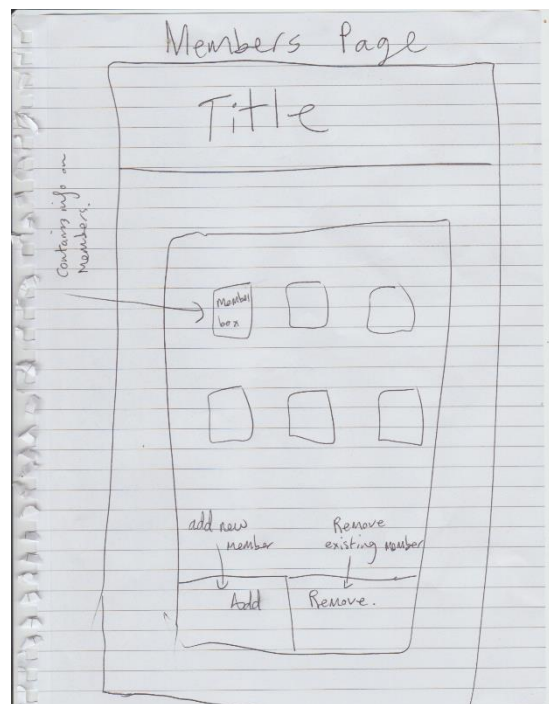
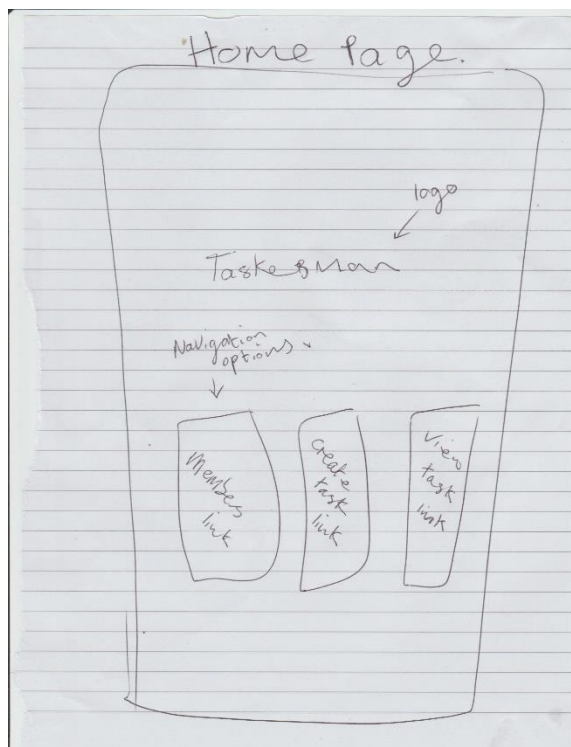
Below is the use case diagram for the user application Tasker CLI.

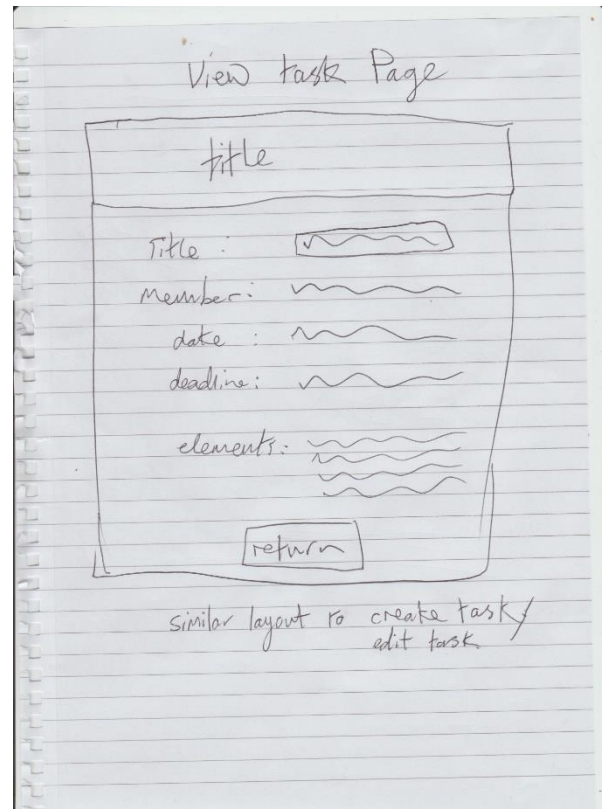
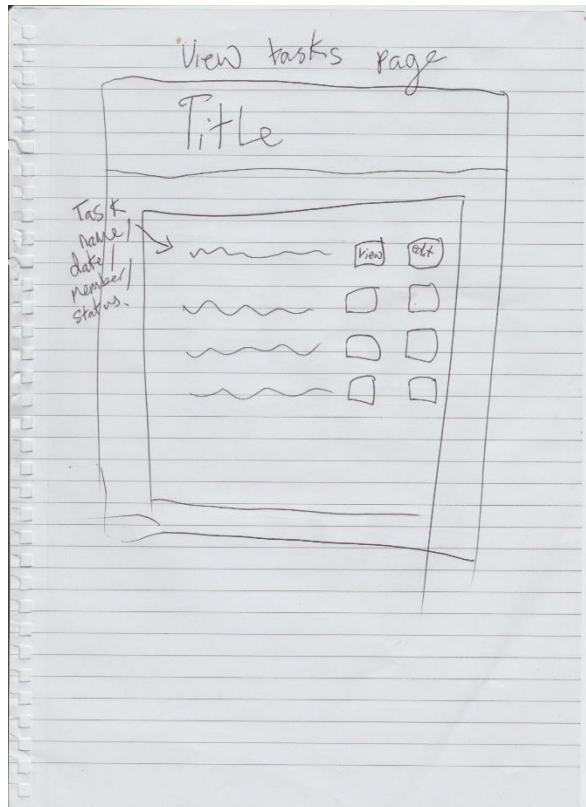


The associations are used in the use case diagram are used to display how the use interacts with the application. You can view all the functionality of the program. So we can see that the application will provides: user identification, the ability to view tasks, to edit the comment about the tasks and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

3.2 User Interface Design

Below are the rough designs drawn up before the final design for the web application Tasker Man.





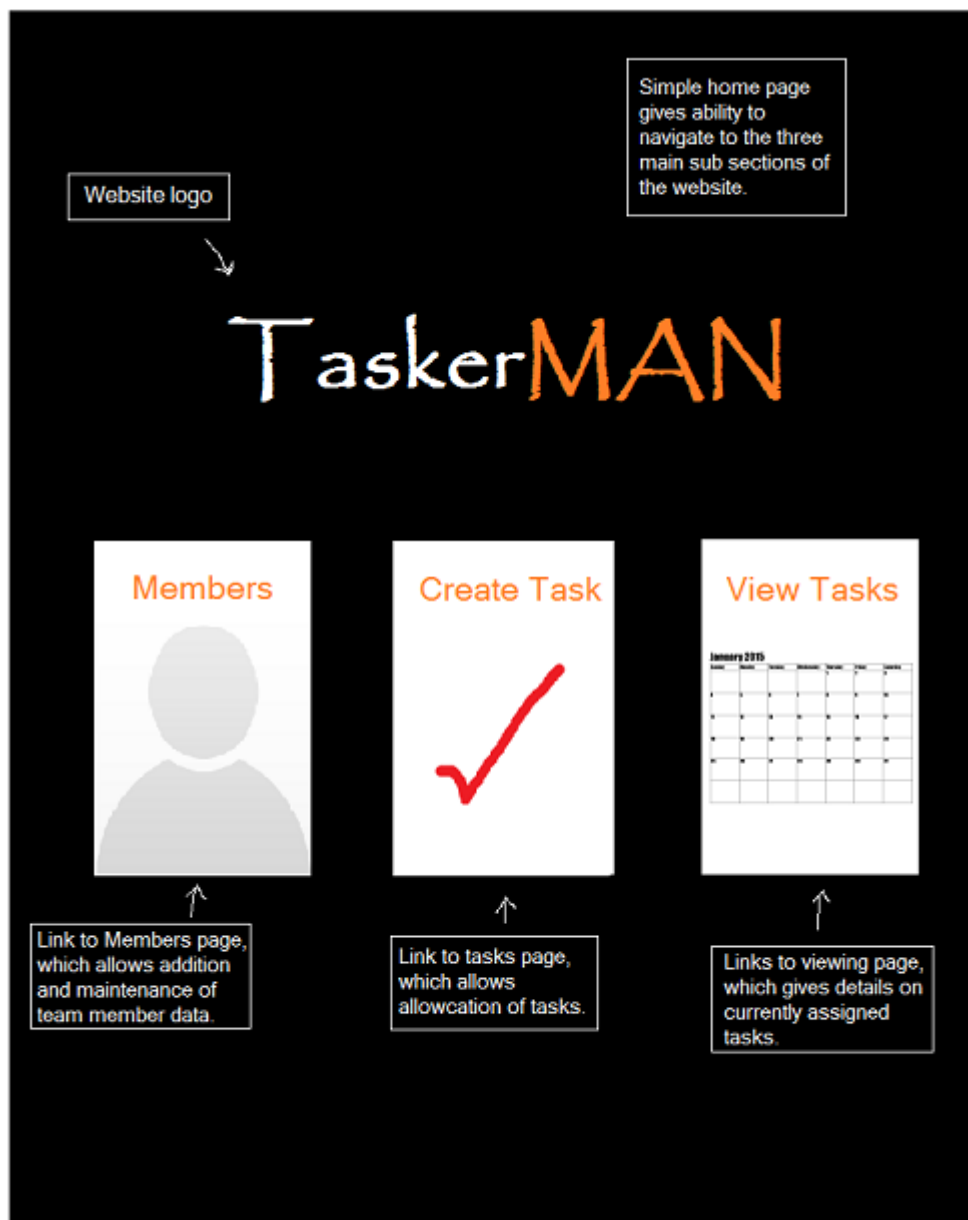
Below are the final designs for the user interface with descriptions.

3.2.1 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 it does not exist. 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

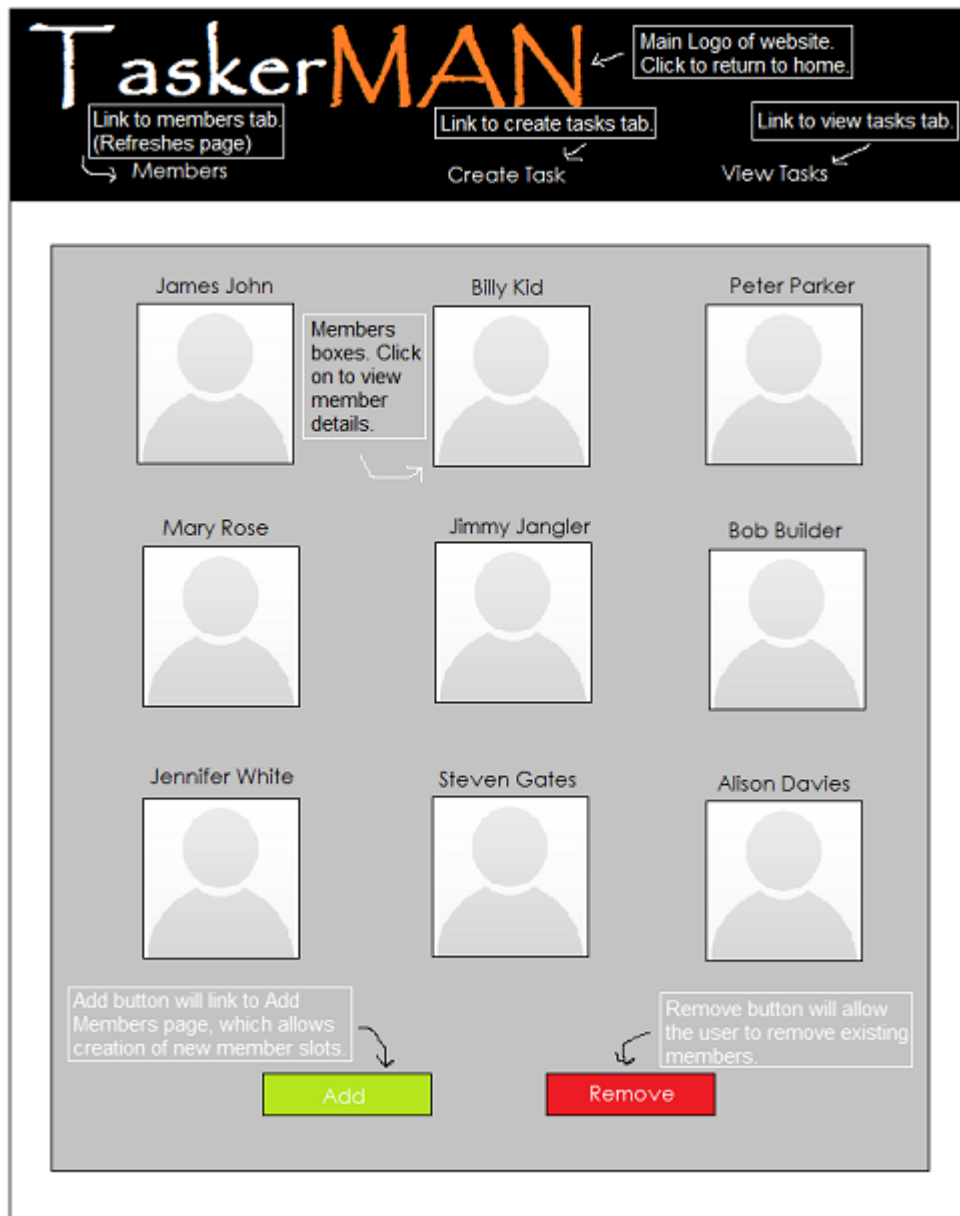
For the home page I've designed a simple page that gives the user the choice to navigate to any of the three main sub sections of the website. Clicking on any of the three boxes will redirect the user to the selected page.



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.



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.



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 screenshot shows the 'TaskerMAN' website interface. At the top is a black navigation bar with the 'TaskerMAN' logo in white and orange. Below the logo are three links: 'Link to members tab.' (labeled 'Members'), 'Link to create tasks tab.' (labeled 'Create Task'), and 'Link to view tasks tab.' (labeled 'View Tasks'). Below the navigation bar is a grey box containing the member's information. On the left is a placeholder for a profile picture, labeled 'Profile picture of member'. To the right of the picture is the text 'Member: James John' with a label 'Members name' pointing to it. Below this are two input fields: 'Full Name:' and 'E-Mail:'. A label 'Submit boxes to enter new details on member' points to both fields. At the bottom of the grey box are two buttons: 'Submit' (labeled 'Submit to save new information') and 'Cancel' (labeled 'Cancel to keep old information').

3.2.6 Create Task Page

The create task page allows the user to create tasks.

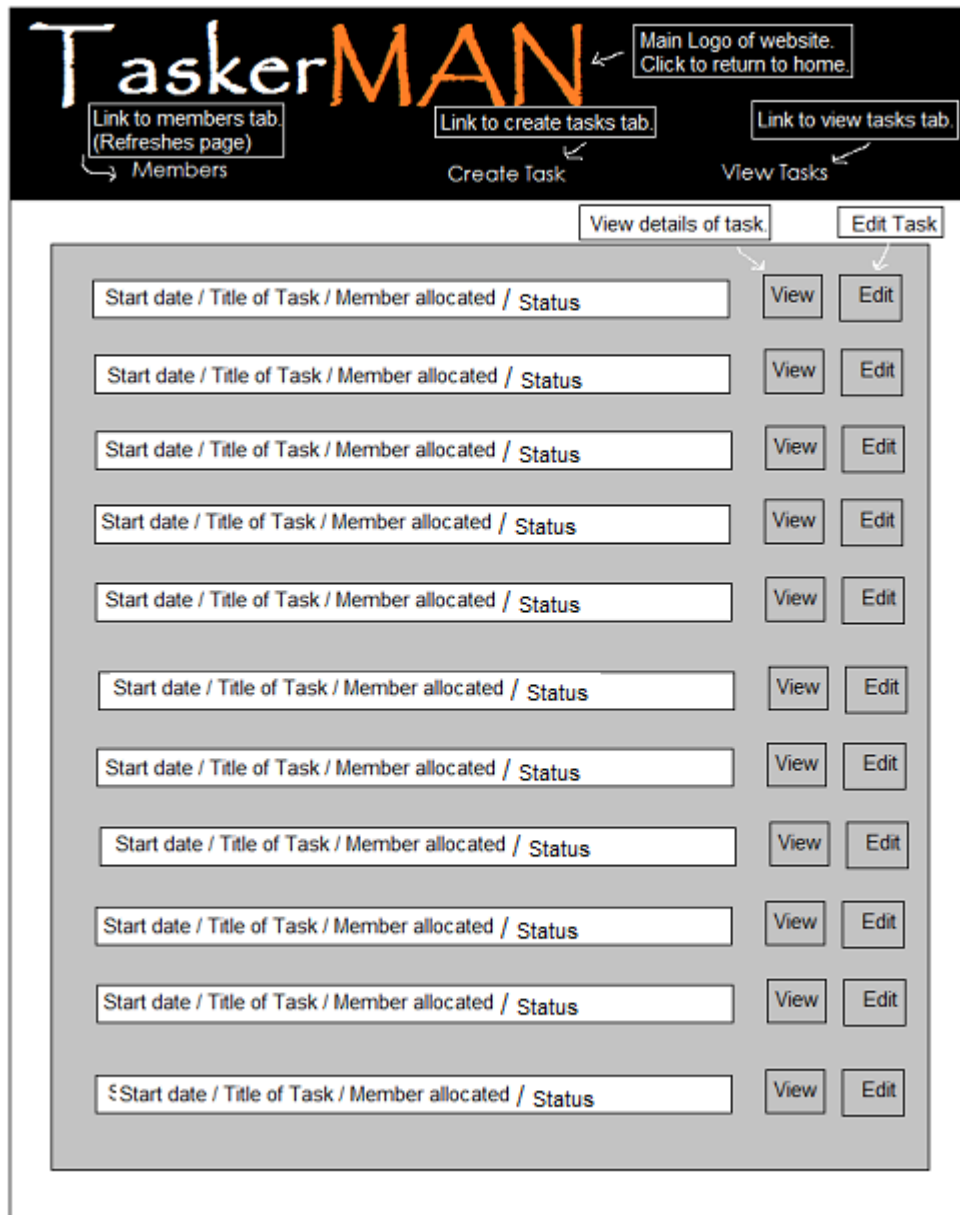
To create a task the site requires information to be submitted by the user: The member to which it is being assigned and various task details (Title of the task, Starting date, Deadline, Task elements). On clicking the drop down list, all available members will be displayed ready for the user to choose 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 screenshot shows the 'TaskerMAN' website interface. The header features the 'TaskerMAN' logo in orange and white. To the right of the logo is a box with the text 'Main Logo of website. Click to return to home.' Below the logo are three links: 'Link to members tab. (Refreshes page)' with a right arrow, 'Link to create tasks tab.' with a right arrow, and 'Link to view tasks tab.' with a right arrow. Below these links are the labels 'Members', 'Create Task', and 'View Tasks' respectively. The main content area is a form for creating a task. It contains the following fields: 'Task title:' with a text input field containing 'Title of task'; 'Allocated Member:' with a dropdown menu showing 'Drop down list to choose member.' and a right arrow; 'Start Date:' with a text input field containing 'Date of beginning of task'; 'Date of completion:' with a text input field containing 'Deadline of task'; and 'Task elements:' with a text area containing 'List of task elements, with at least one member' and a detailed description: 'A task element is free text used to describe a step in the task, and associated free text for comments added by the member carrying out the task to report on progress with the step.' At the bottom of the form are two buttons: a green 'Submit' button and a red 'Cancel' button.

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.



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 screenshot shows the 'TaskerMAN' website interface. The header is black with the logo 'TaskerMAN' in white and orange. Below the logo are three links: 'Members', 'Create Task', and 'View Tasks'. The main content area is a light gray box with a white border. It contains the following fields:

- Task title:** A text input field with the placeholder text 'Title of task'.
- Allocated Member:** A text input field with the placeholder text 'Members name'.
- Start Date:** A text input field with the placeholder text 'Date of beggining of task'.
- Date of completion:** A text input field with the placeholder text 'Deadline of task'.
- Task elements:** A text area with the placeholder text 'List of task elements, with at least one member'. Below the placeholder text is a description: 'A task element is free text used to describe a step in the task, and associated free text for comments added by the member carrying out the task to report on progress with the step.'

At the bottom of the main content area is a blue button with the text 'Return to Tasks'.

Annotations on the screenshot include:

- A box pointing to the 'TaskerMAN' logo: 'Main Logo of website. Click to return to home.'
- A box pointing to the 'Members' link: 'Link to members tab. (Refreshes page)'.
- A box pointing to the 'Create Task' link: 'Link to create tasks tab.'
- A box pointing to the 'View Tasks' link: 'Link to view tasks tab.'

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.

TaskerMAN

Link to members tab. (Refreshes page) → Members

Link to create tasks tab. → Create Task

Link to view tasks tab. → View Tasks

Main Logo of website. Click to return to home.

Task title: Title of task

Allocated Member: Drop down list to choose member.

Start Date: Date of beginning of task

Date of completion: Deadline of task

Task elements: List of task elements, with at least one member

A task element is free text used to describe a step in the task, and associated free text for comments added by the member carrying out the task to report on progress with the step.

Status: ☐ Allocated ☐ Abandoned ☐ Complete

Status of current task

Submit Cancel

Below are the rough designs for the user application Tasker CLI.

A hand-drawn sketch of a login form. At the top center is a rectangular box containing the text "Tasker Logo". Below this, on the left side, are the labels "username" and "Password". To the right of "username" is a single-line text input field. To the right of "Password" is a single-line text input field. Below the password field is a two-column table with the headers "login" and "offline".

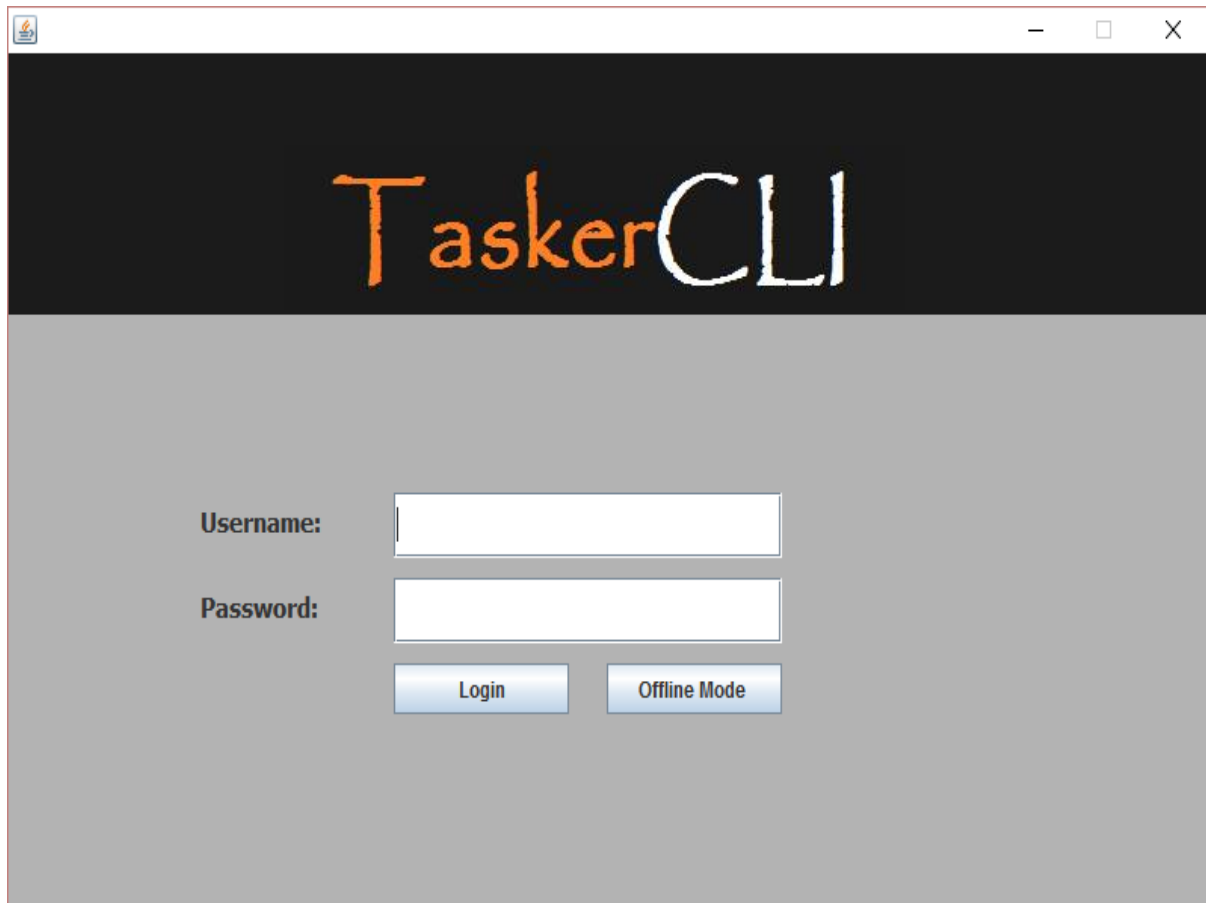
A hand-drawn sketch of a main interface. At the top left is a box labeled "current user". To its right is a box labeled "Search field". Below "current user" is a large rectangular area labeled "JText" with the text "Here you can edit the selected task discription" (note the misspelling of "description"). Below this area is a button labeled "edit task". To the right of the "JText" area is a large rectangular area labeled "JTable - Populated by the database".

A hand-drawn sketch of a task editor. At the top left is a box labeled "current task". To its right are two radio buttons. The first radio button is labeled "completed" with an arrow pointing to it. The second radio button is labeled "on-going (default)" with an arrow pointing to it. A bracket to the right of these two radio buttons is labeled "radio buttons". Below the "current task" box is a large rectangular area labeled "Current task, details editor". At the bottom center is a button labeled "Submit".

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

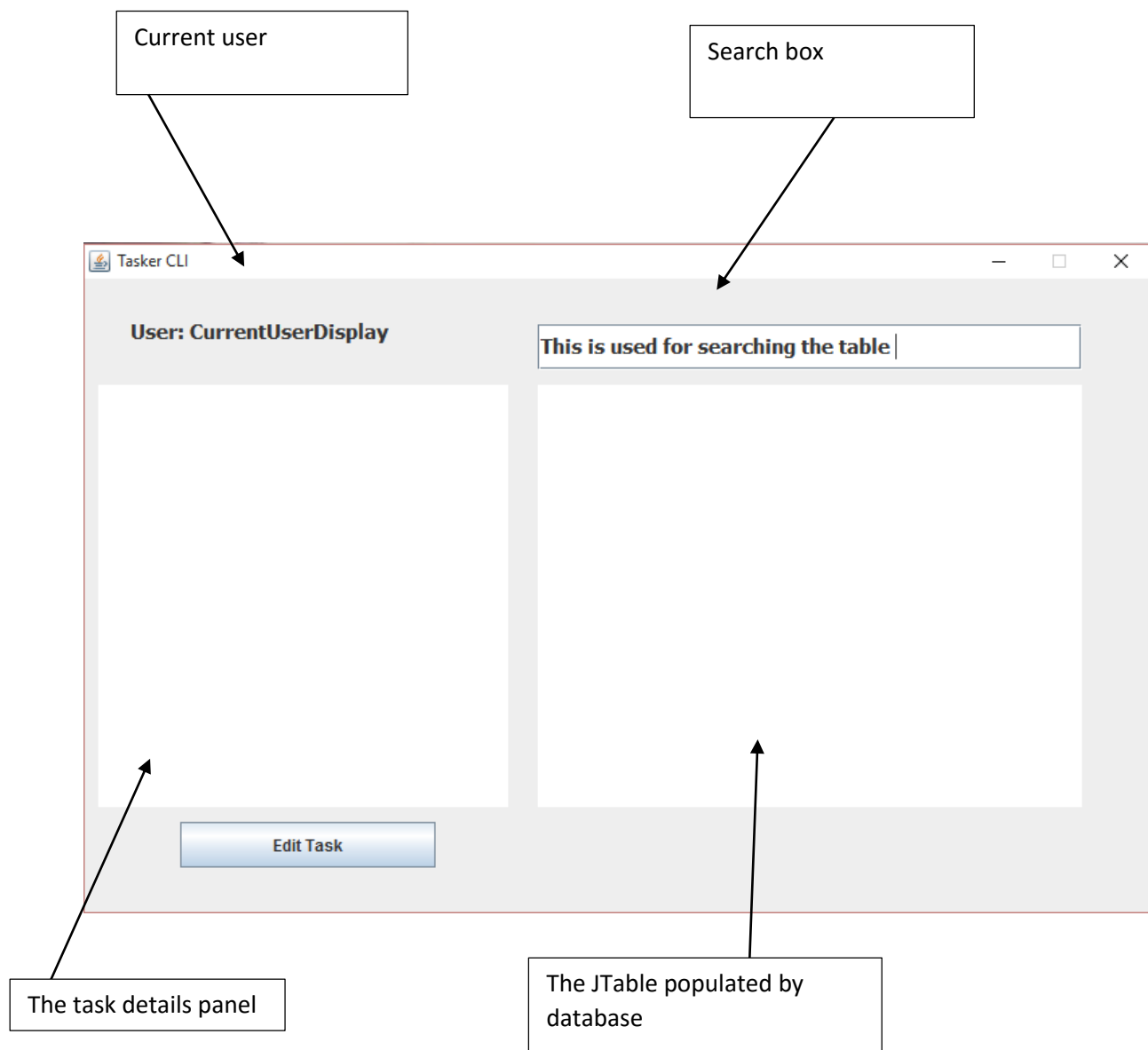
3.2.10 Login page

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.



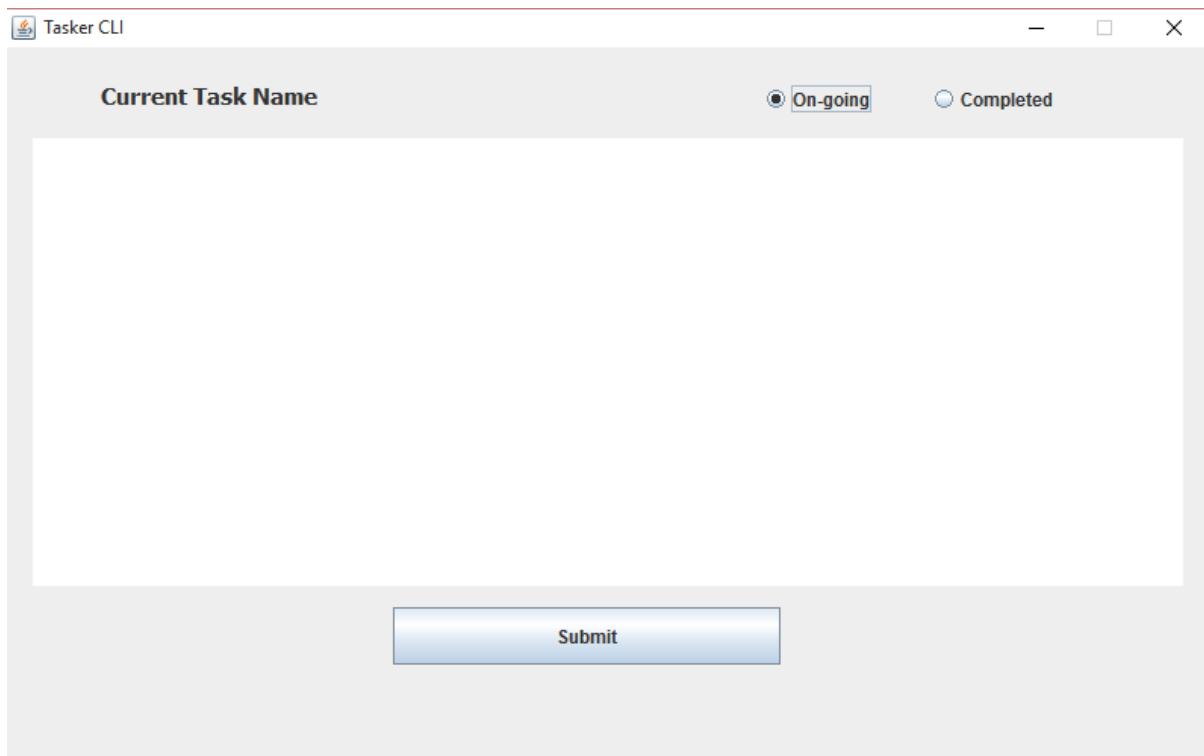
3.2.11 User Application

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.



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.



The screenshot shows a window titled "Tasker CLI" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a light gray background. At the top, there is a label "Current Task Name" on the left. To its right are two radio buttons: "On-going" (which is selected, indicated by a filled circle) and "Completed" (which is unselected, indicated by an empty circle). Below these elements is a large, empty white rectangular area, likely a text input field for the task details. At the bottom center of the window is a blue button with the text "Submit".

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 accordingly. 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 be 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 its full description will be displayed next to it in a panel. The user also has the ability to search the table, to help find a specific task. Once a task has been selected the user can press a button below the full task description to edit the task.

Dependency: TaskerLogin.java

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



5.0 Detailed Design

5.1 Schema Design

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

members { id, name, email, password }

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.

5.2 Format of Data Transmitted

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.

5.3 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

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