Final System Design Specification

**GROUP 10**

|  |  |
| --- | --- |
| Author: | Curtis Griffiths |
| Config Ref: | SE\_10\_2\_1 |
| Date: | 2015-10-17 |
| Version: | 1.7 |
| Status: | Release |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2015

CONTENTS

[CONTENTS 2](#_Toc433811343)

[1. Introduction 3](#_Toc433811344)

[1.1 Purpose of this Document 3](#_Toc433811345)

[1.2 Scope 3](#_Toc433811346)

[1.3 Objectives 3](#_Toc433811347)

[2. Deployment DESCRIPTION 3](#_Toc433811348)

[2.1 Applications in the system 3](#_Toc433811349)

[2.1.1 TaskerSRV 4](#_Toc433811350)

[2.1.2 TaskerMAN 4](#_Toc433811351)

[2.1.3 TaskerCLI 4](#_Toc433811352)

[2.2 Application Interactions 5](#_Toc433811353)

[3. Interaction DESIGN 6](#_Toc433811354)

[3.1 Use-cases 6](#_Toc433811355)

[3.2 User Interface Designs 8](#_Toc433811356)

4. COMPONENT DESCRIPTION…………………………………………………………………..16

5. SIGNIFICANT CLASSES………………………………………………………………………...17

5.1 Interface Description………………………………………………………………………..17

6. DETAILED DESIGN……………………………………………………………………………...18

6.1 UML Sequence Diagram……………………………………………………………………18

6.2 State Diagram……………………………………………………………………………….19

6.3 Activity Diagram……………………………………………………………………………19

6.4 Significant Data Structures………………………………………………………………….20

[REFERENCES…………………………………………………………………………………………..21](#_Toc433811357)

[DOCUMENT HISTORY …22](#_Toc433811358)

# Introduction

## Purpose of this Document

This document will provide the design specification that will be used by the group to ultimately implement the system required. The document will use a variety of different designs and diagrams to give the group as accurate information as possible, for the end system to be as efficient as possible. The document will cover all areas of the system, and describe the different components and how they will be implemented at a high level, this will consist of many different designs, from UML diagrams, to interface designs to clearly outline how the group intend on implementing the system, and what it will go on to achieve. Due to the level of detail this document will delve into, it will therefore be a lengthy document, with many sections covering the topics that are required to be discussed to produce the best quality of product possible.

## Scope

The scope of this document will be very broad in terms of what the content will go on to cover, it will consist of five main sections:

* Deployment Description
* Interaction Design
* Component Design
* Significant Classes
* Detailed Design

Each section is completely relevant to being able to produce the most effect of end products. Different sections of the document are aimed at different components that will make up the end system, detail is required to be inputted for every component, ensuring every component is built to a high standard. The document is required to be read only by the people that are involved with the project group, as it is important for everyone that is included within the group to have awareness to what the system is, and how it will be implemented.

## Objectives

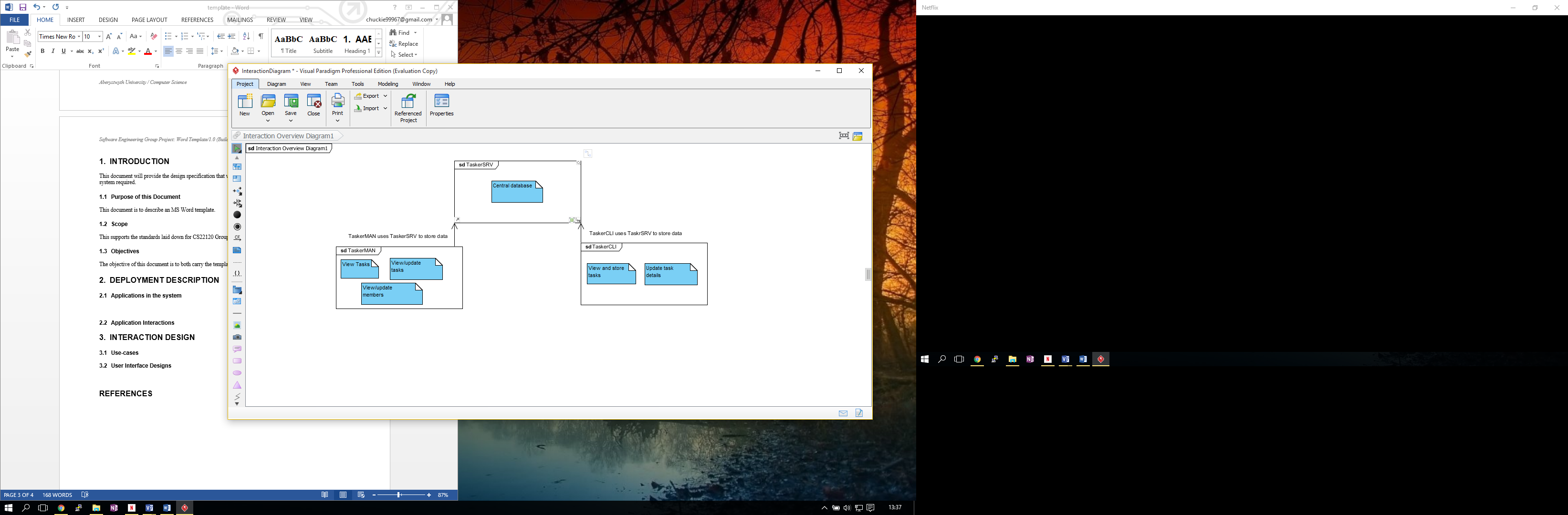
The main, primary objective of the design specification document is to provide the members of the group how to explain and describe how the system should be implemented. The objectives of this document are:

* Provide the group with a detailed description of what makes up the system.
* Provide details on how the different components of the system will be built.
* Specifics regarding how the components of the system will interact.
* Supply both interface and detailed code designs giving a full idea on how the components will work.

# Deployment DESCRIPTION

## Applications in the system

This first section of the design document, the deployment description is mainly aimed at what makes up the system, the separate applications involved that will interact to provide the overall system. The specification requires there to be a CLI (TaskerCLI) and a web (TaskerMAN) interfaces that the user is able to use, there already there are two separate applications that will be a part of the system, with both of them having different purposes in the system. There will also be one more application that will complete the system and will be the central application that the CLI and the web page will use to function, this will be known as the TaskerSRV, and will provide the database that will store, centrally, all of the detail that the applications will need to act as the task management system. The diagram below simply shows how the system is built up, showing the three applications involved that are listed above, as well as showing a simple version of the interactions between the applications (which is discussed further in the next section 2.2). As it shows, TaskerMAN and TaskerCLI will work separately from one another, and will both only rely on the TaskerSRV to store the information.



### TaskerSRV

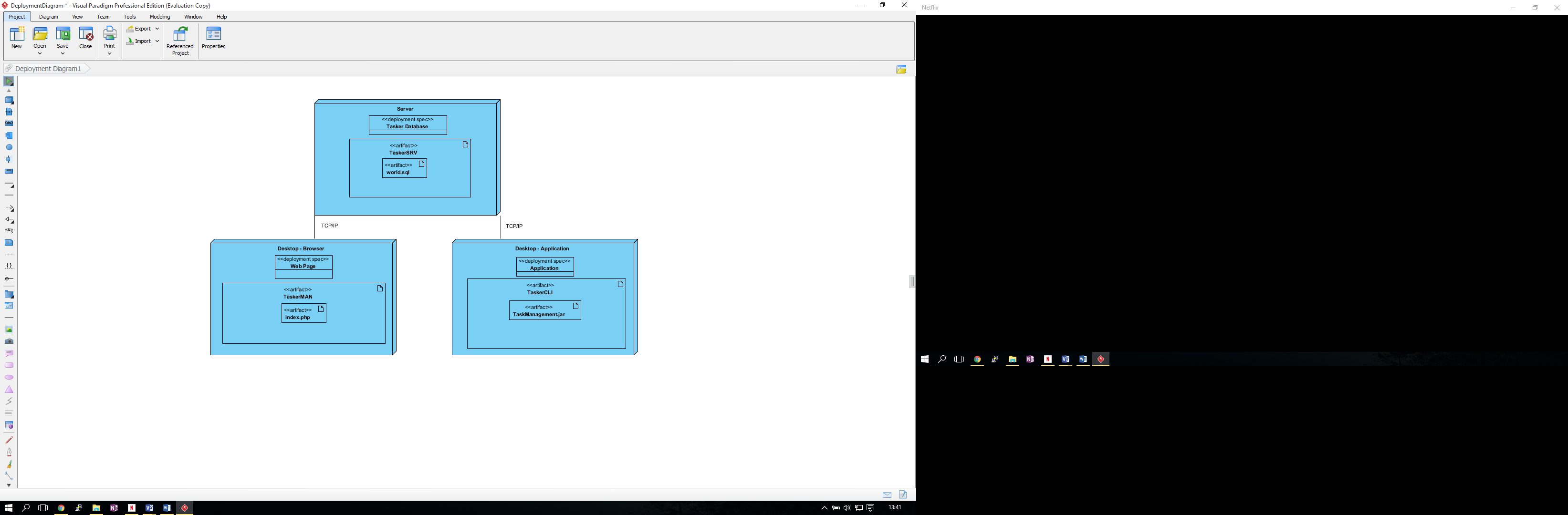
As shown in the diagram above and briefly described, the TaskerSRV is on the three applications used in the system, this will provide the backbone storage for the remaining two applications, as well as being the applications that the users won’t see or use (directly), only the implementers of the system will use the TaskerSRV as the database will be used by the two applications to store data. Although this component of the system won’t be directly used by any member of the company, be it an Admin member or just an employee, it is ultimately the most crucial component to the system as it provides the storage for all the information that allows the task management system to be able to run.

### TaskerMAN

The TaskerMAN will provide a web based interface that should be accessible from and machines online, this will be the main application that will be used by the employees, it will provide the majority of the functionality that the system will proved, as shown in the diagram above, the overall functionality is to be able to view the tasks, update the tasks, and finally the admin staff must be able to edit the team members information, as well as it being shown. All of the information that the website will use and manipulate will be stored in the TaskerSRV that the TaskerMAN website will use. Because the TaskerMAN component will be a web interface, the users will have to access it through the use of a web browser, from a given URL that will provide the interface for the site. The TaskerMAN will actually be hosted on the server that the TaskerSRV is on, but the HTML client side will use the PHP server side to access the database on TaskerSRV.

### TaskerCLI

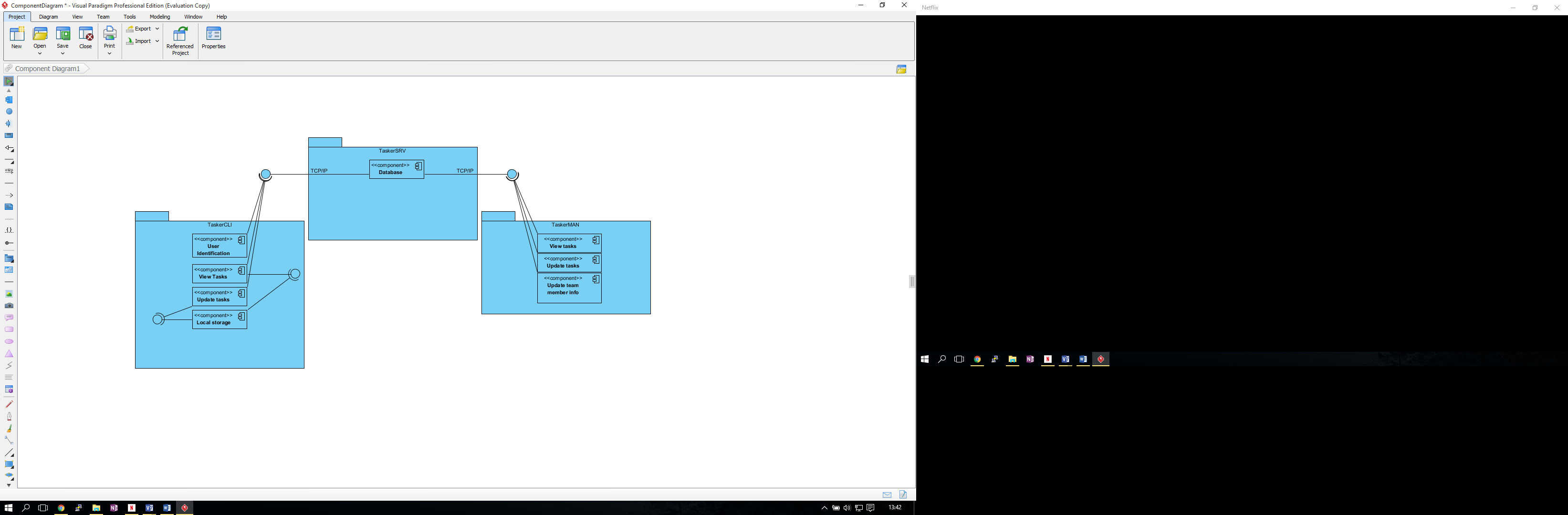
The final component of the system to discuss is the desktop application that will be provided through TaskerCLI, it will be a runnable application that users can run from their machine desktops (provided they meet the requirements which will come later on in the document). The TaskerCLI component will provide similar functionality to the TaskerMAN component, just not as extensive, as the only functionality will be to view and update the tasks, according to a specific user, also, the application must allow for the local storage of the tasks from the database on TaskerSRV so that this component can work offline, and then the synchronisation when the device is back online again. This application will be built using the Java language, and will provide the user with a conventional user interface, similar to the website in TaskerMAN, providing the user with an alternative method, with its main purpose being that the employee is able to use the application, and update tasks offline, furthering the efficiency of the system.



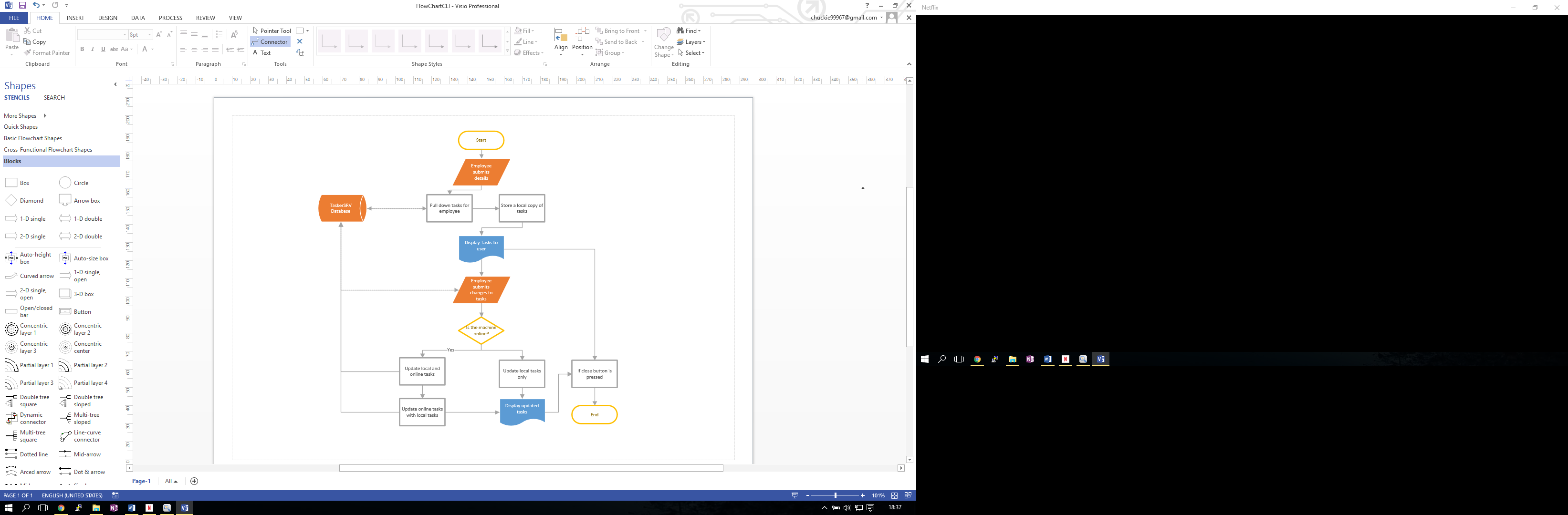
This deployment diagram shows in a bit more detail than the previous diagram what is involves in the three different components, and what makes them up, such as stating that the TaskerCLI desktop application will be runnable through the use of a .jar file, and that the web page will be centred around the index.php file, that provides both the client side and server side of functionality to the component. The diagram clearly, and simply outlines the separate applications that will be combined to produce the overall system.

## Application Interactions

Now that the components, and separate applications in the system have been identified, the next stage for the design specification is to provide a design on the interactions between the components. The following component diagram is similar to the diagrams above in section 2.1, however, providing further detail to how the different aspects of the different components will interact with each other.



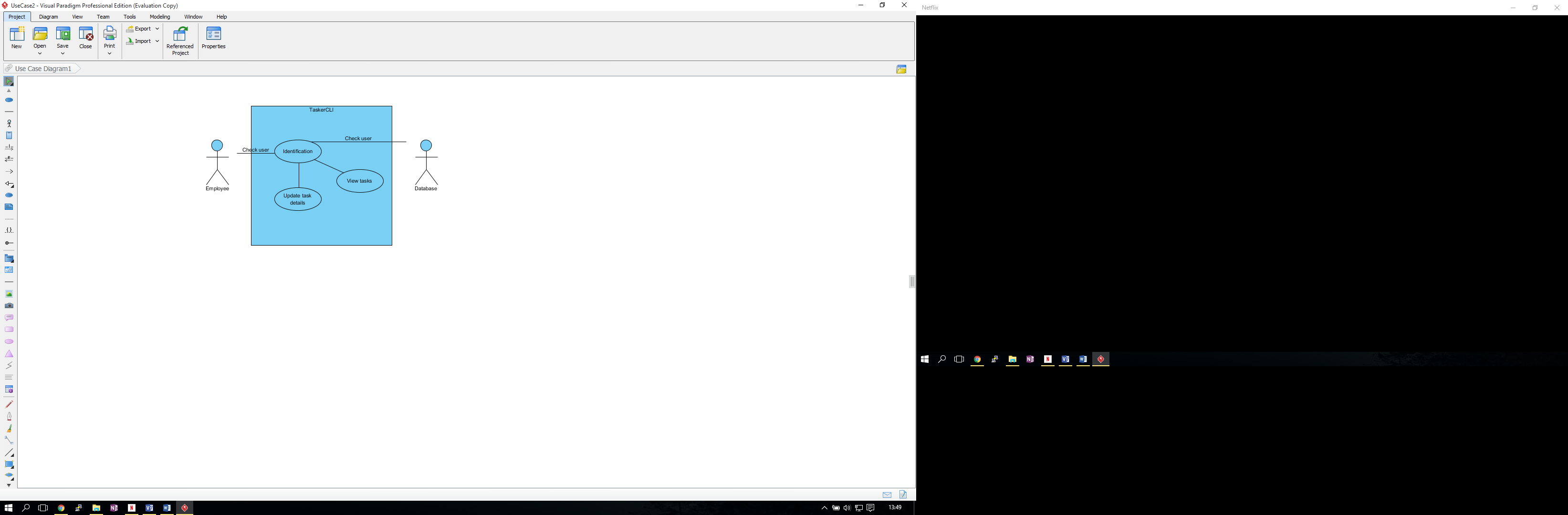
In the diagram above, the three different components have components within them that describe the functionality that the applications will provide, for example, displaying the data to the user, this diagram shows what component of the different applications will use the other applications. The only interactions between the applications that will happen within the system are the ones shown in the diagram, and they are the interactions between the TaskerCLI and TaskerMAN applications interacting with the TaskerSRV interface in order to manipulate the data in the database to allow the complete system to work, other than that, there are no other interactions between any of the applications that make up the system. The calls that are used in order to interact with the TaskerSRV database to the desktop applications and the website is TCP/IP on both occasions to pass the queries to the database, and pass the information back to the application,

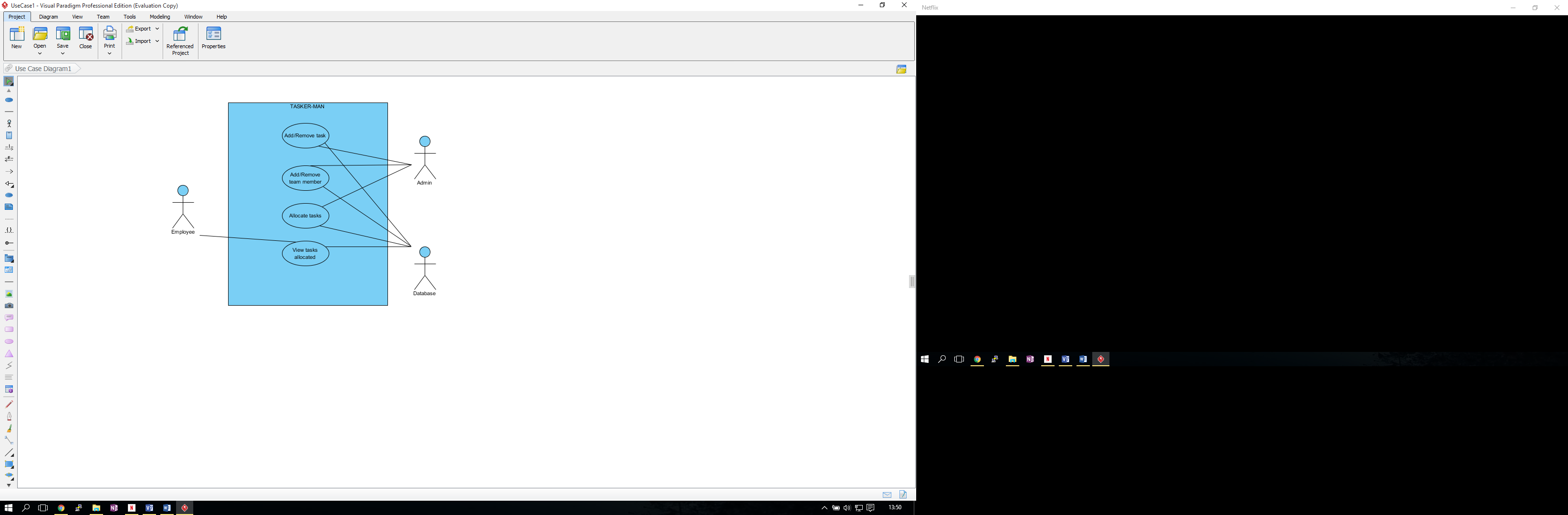


An example of the interactions are shown above in the flowchart shows the flow of the TaskerCLI application, and what functions of the application are required to interact with another application within the system. As is visible, there are four occasions where the TaskerCLI application will need to interact with the database, the database is shown in the flowchart on the far left, and the four occasions where the application is required to use the database are shown with linked connections and arrows pointing to and from the database.

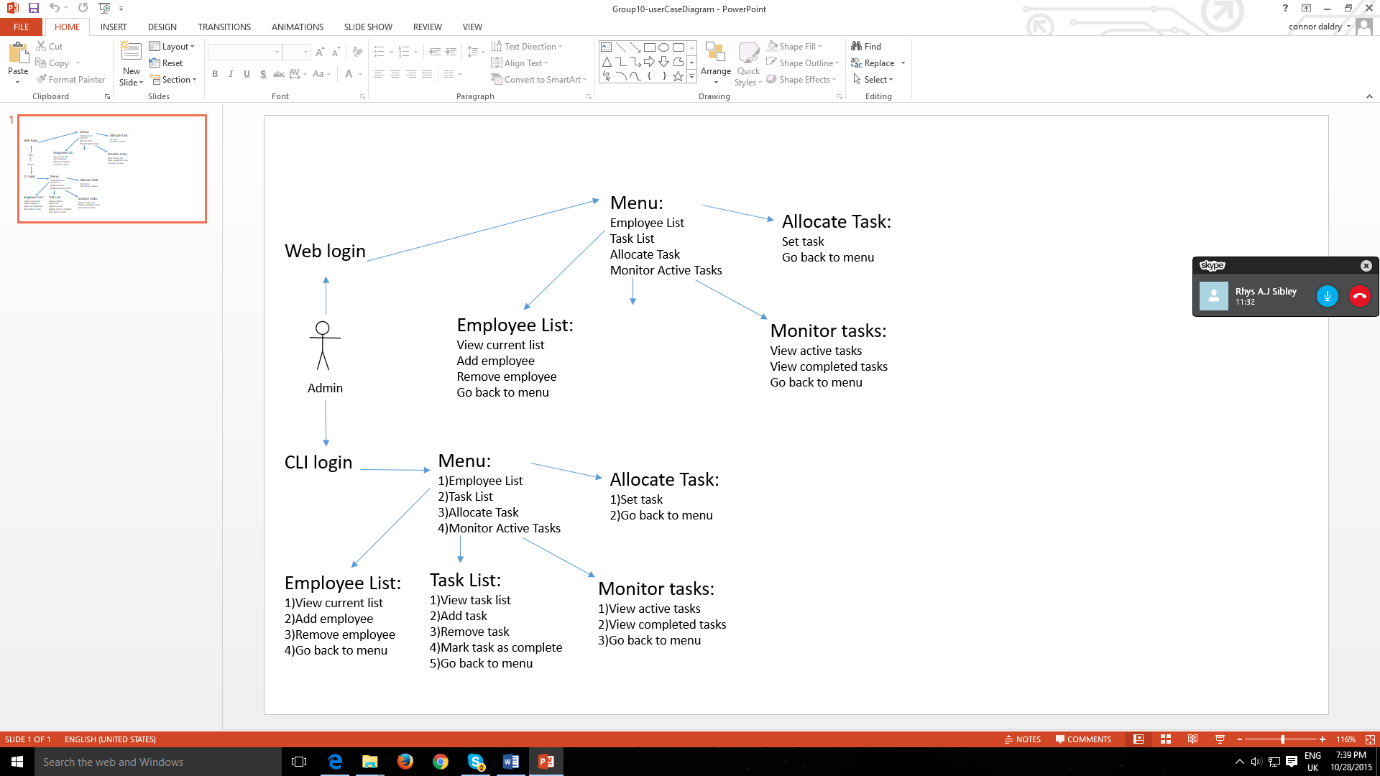
# Interaction DESIGN

## Use-cases





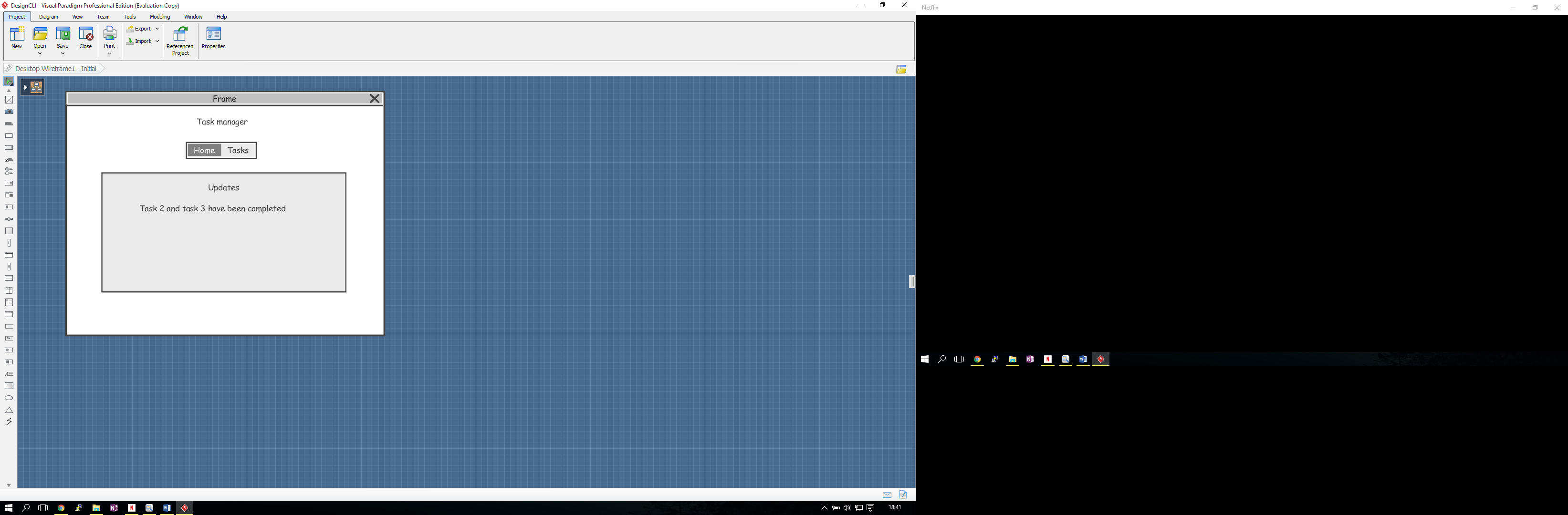
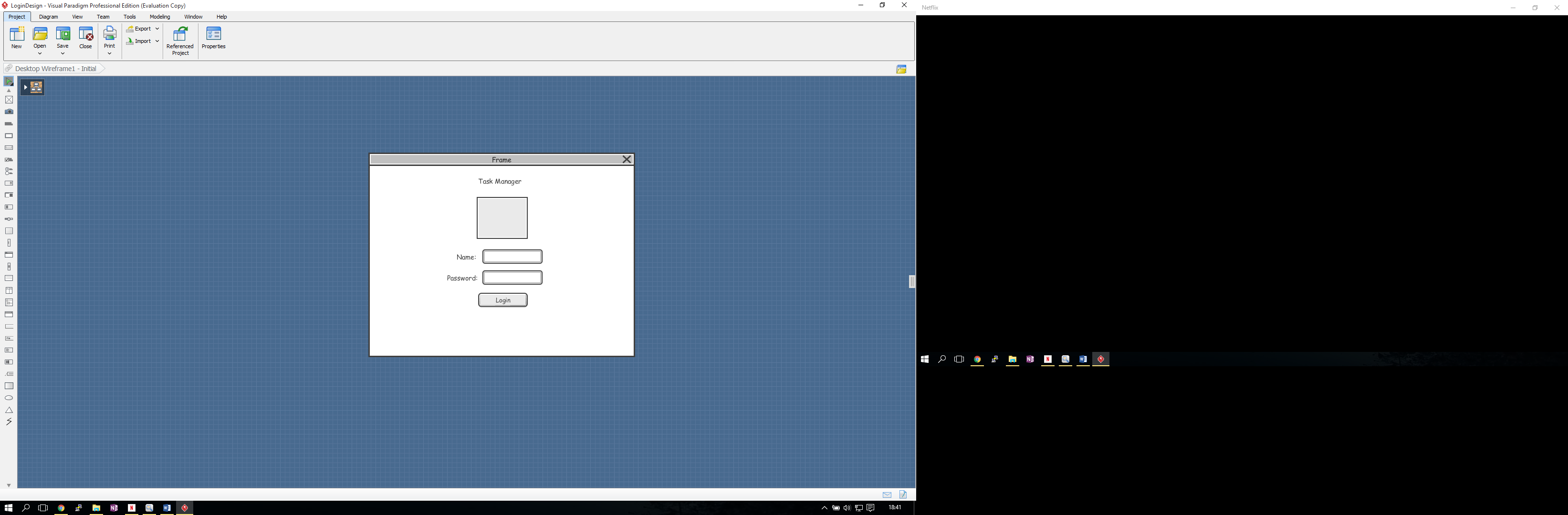
Above you can see how both users for this project will interact with the final product, there are only 2 due to the fact in reality only the employees and the admin/manager will have any need to access the database other users are not necessary and as such won’t have access. The employee however will only have very limited access, in fact their entire view of the product will be viewing their tasks and not have any editing options except to say their task is completed and to update their progress in a task. The admin/manager however has full access to the product and their entire job involves assigning tasks, removing obsolete tasks and adding new tasks as it becomes relevant.

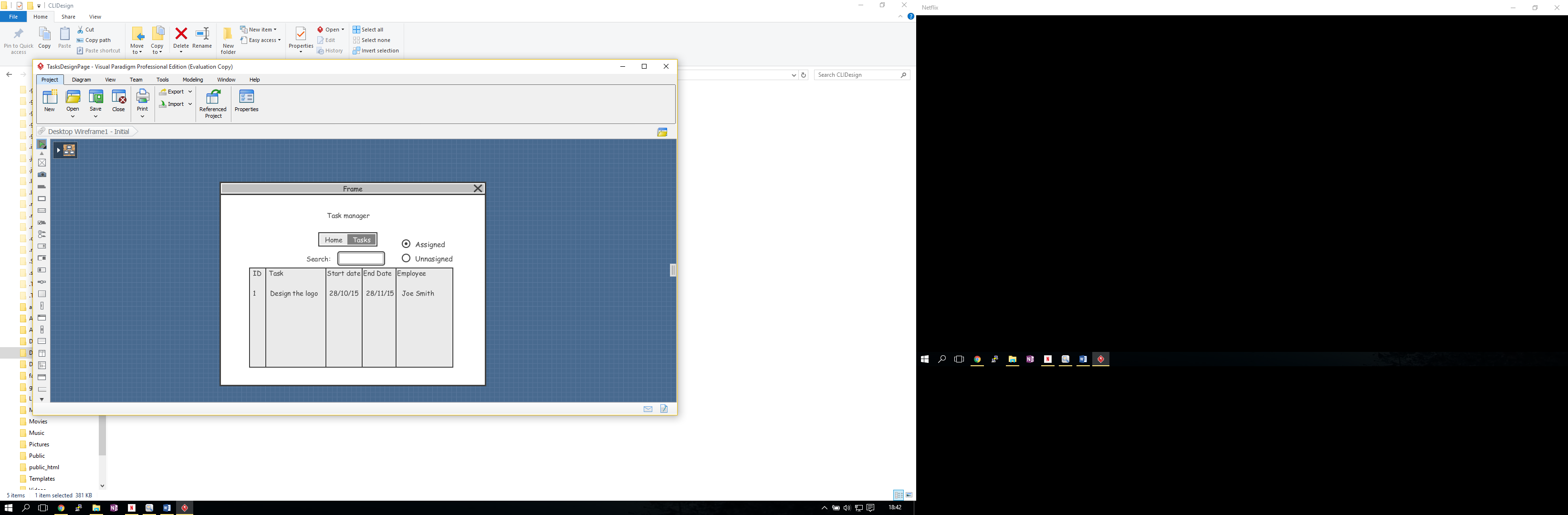


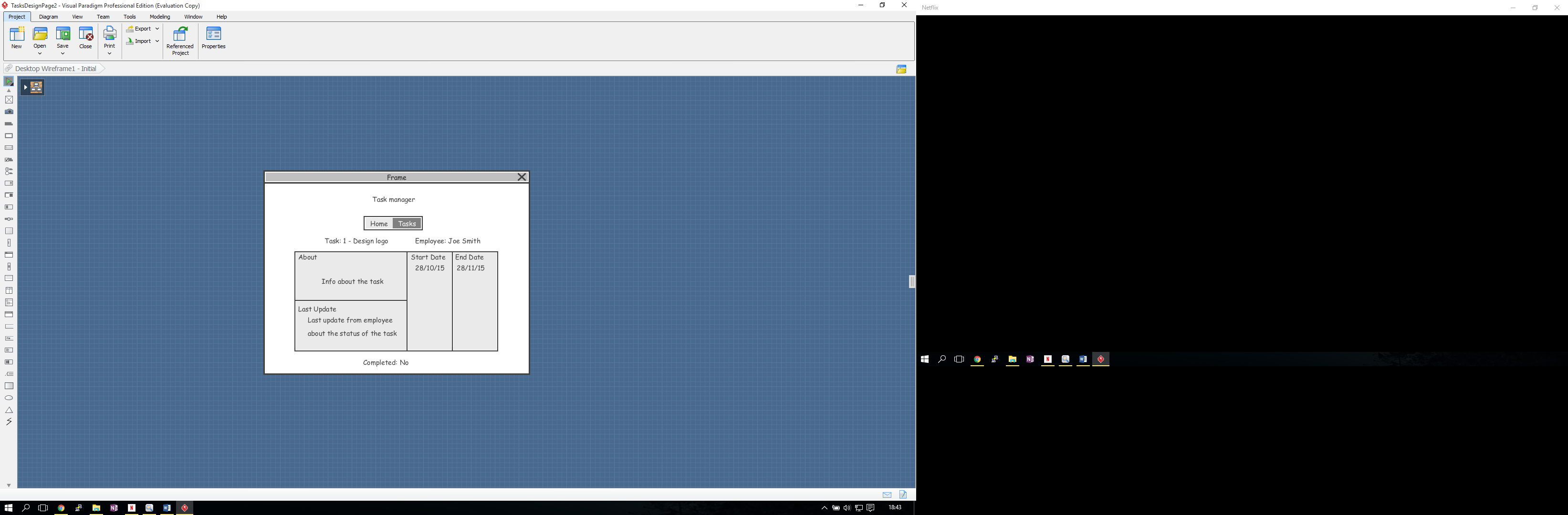
As you can see from above the 2 ways the user can access the product are from a web login screen and from a command line interface (CLI) which acts more like a backup in case the internet connection is lost, both layouts are very similar except CLI in which there will be no graphical information but rather simply pure data, some users may prefer this as it is much simpler to execute and go through stages without getting confused.

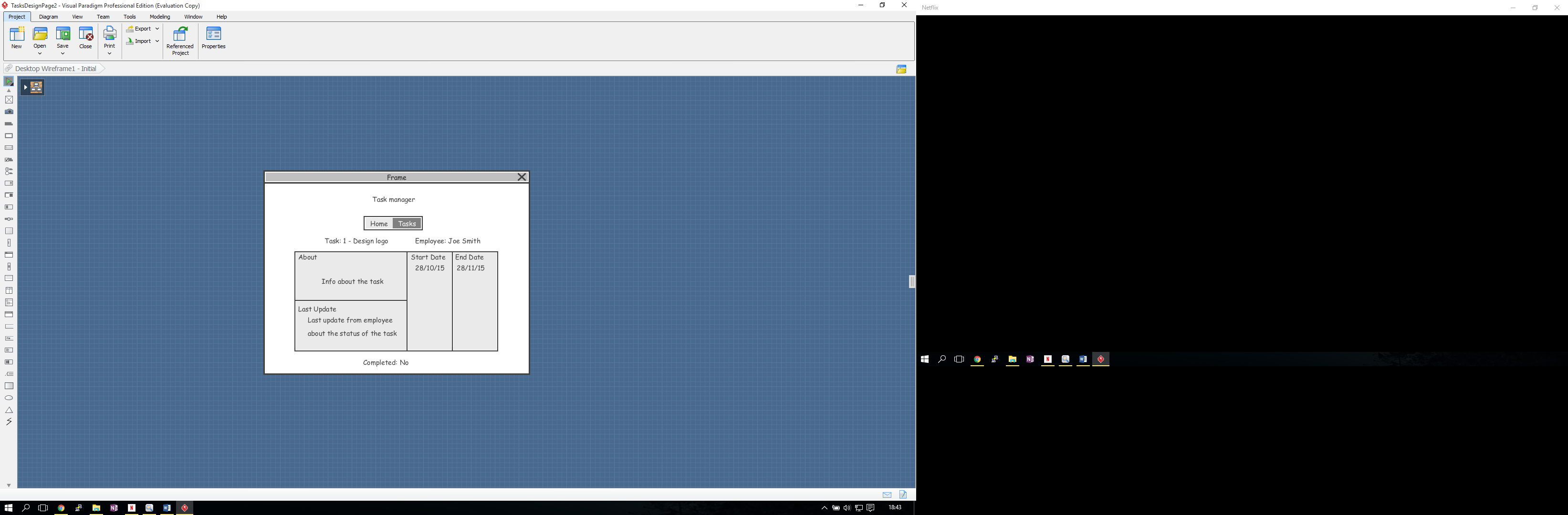
## User Interface Designs

* **TaskerCLI**



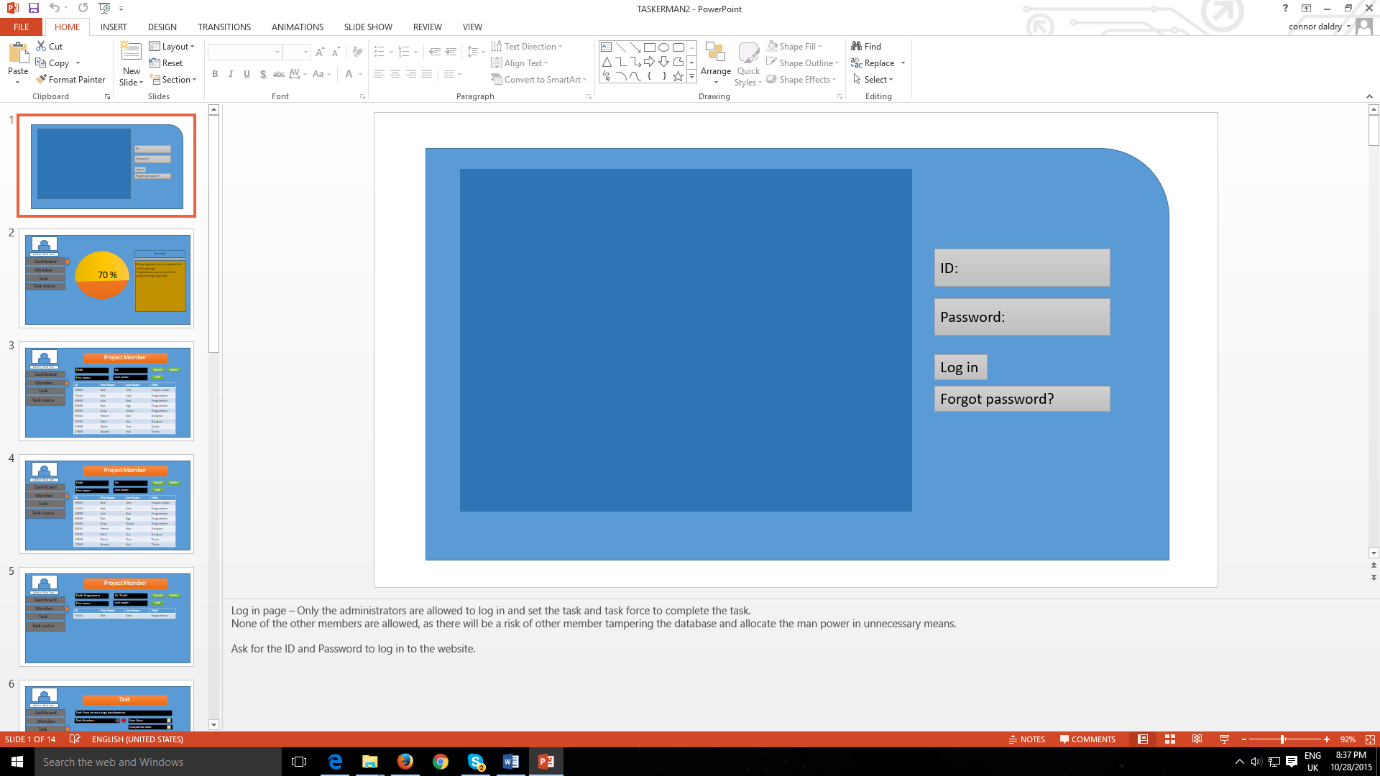






Above you see the proposed design for the web login from the viewpoint of an employee, following the CLI example the design is very minimal to prevent confusion and increase the speed at which users may access the system, the first screen shows the web login screen which prevents unwanted users from accessing certain features such as admin permissions or simply prevent them from entering the system if they are not even employees. Screen 2 above shows the employee’s only accessible page, from the start they see the overview of all tasks completed currently, to see more detail they need to click the Tasks button located next to the Home button. When the Tasks button is selected more information on all tasks given to that individual are displayed including the start date and end date, information on task and any updates given down from the manager.

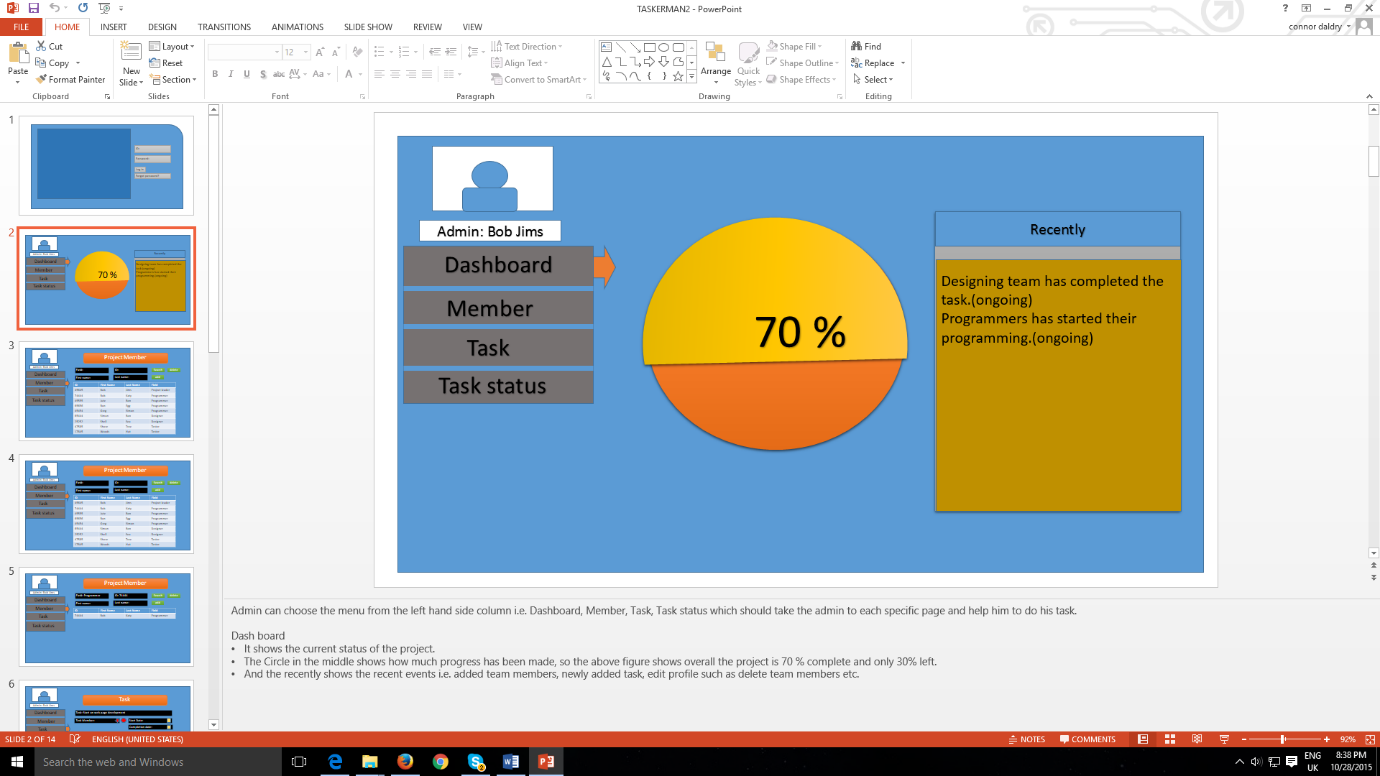
* **TaskerMAN**



Log in page – Only the administrators are allowed to log in and set the task and task force to complete the task.

None of the other members are allowed, as there will be a risk of other member tampering the database and allocate the man power in unnecessary means.

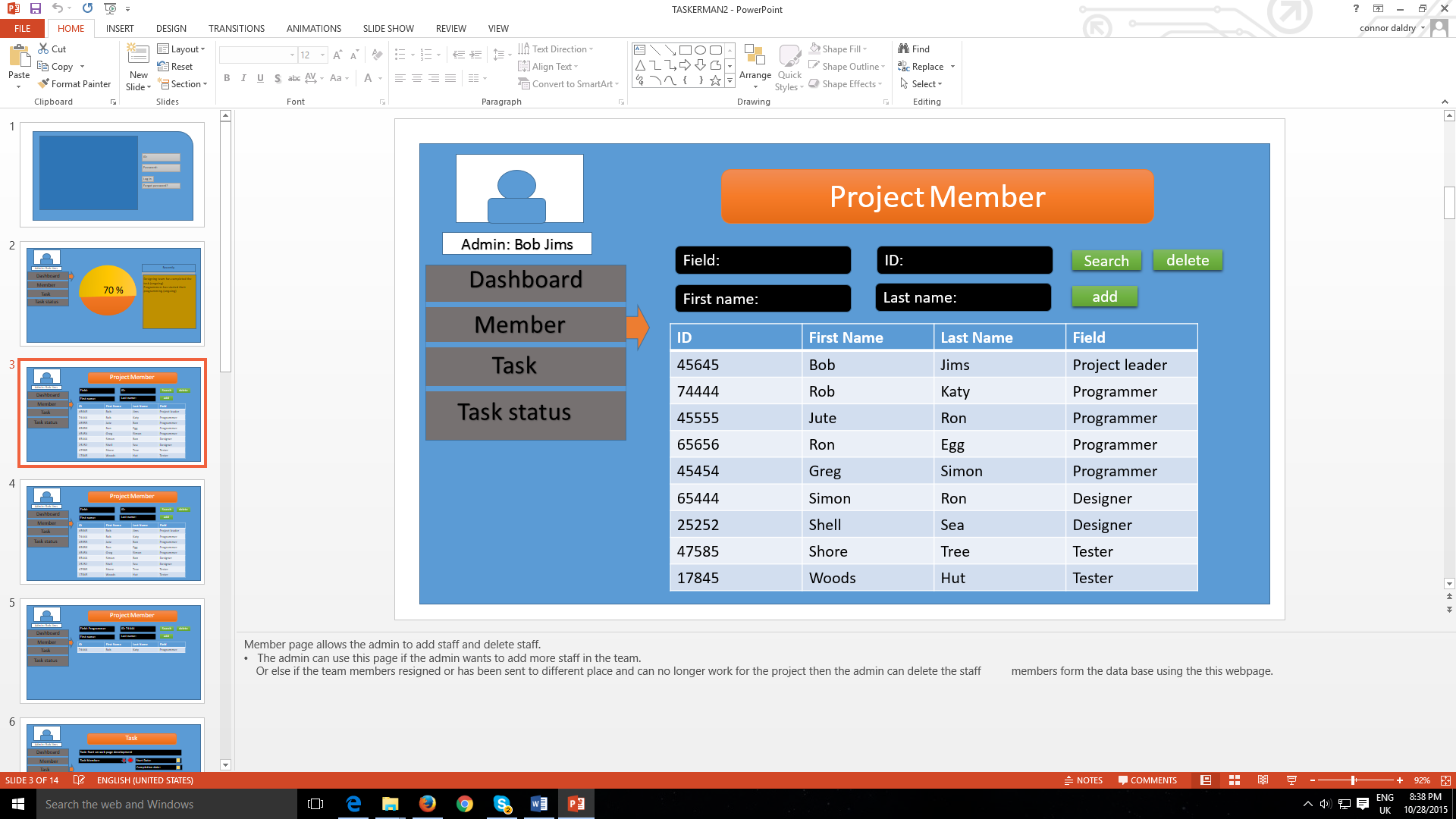
Ask for the ID and Password to log in to the website.



Admin can choose the menu from the left hand side column i.e. Dashboard, Member, Task, Task status which should take the admin to each specific page and help him to do his task.

Dash board

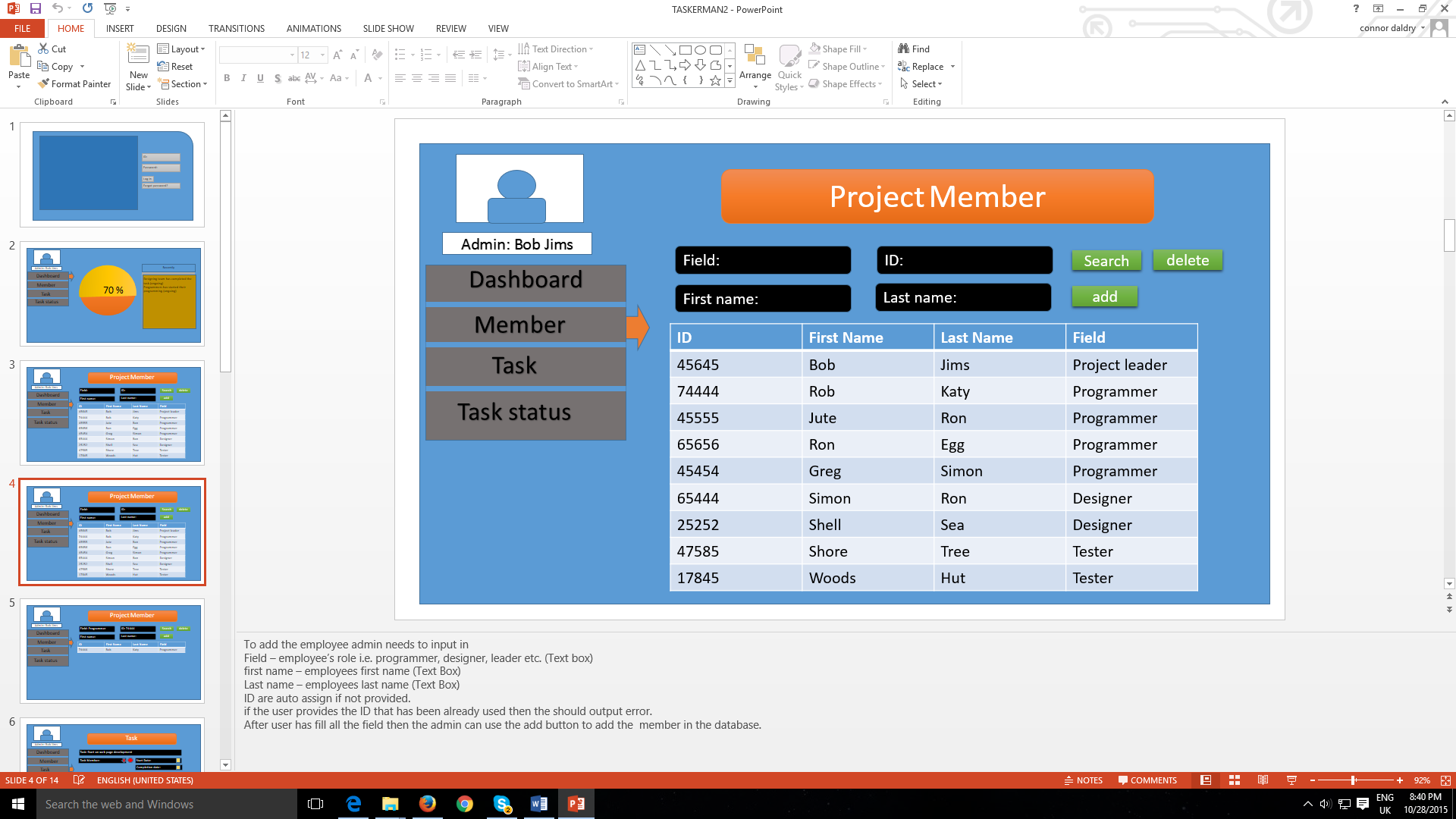
* It shows the current status of the project.
* The Circle in the middle shows how much progress has been made, so the above figure shows overall the project is 70 % complete and only 30% left.
* And the recently shows the recent events i.e. added team members, newly added task, edit profile such as delete team members etc.



Member page allows the admin to add staff and delete staff.

* The admin can use this page if the admin wants to add more staff in the team.

Or else if the team members resigned or has been sent to different place and can no longer work for the project then the admin can delete the staff, members form the data base using the this webpage.



To add the employee admin needs to input in

Field – employee’s role i.e. programmer, designer, leader etc. (Text box)

First name – employees’ first name (Text Box)

Last name – employees last name (Text Box)

ID are auto assign if not provided.

If the user provides the ID that has been already used then they should output error.

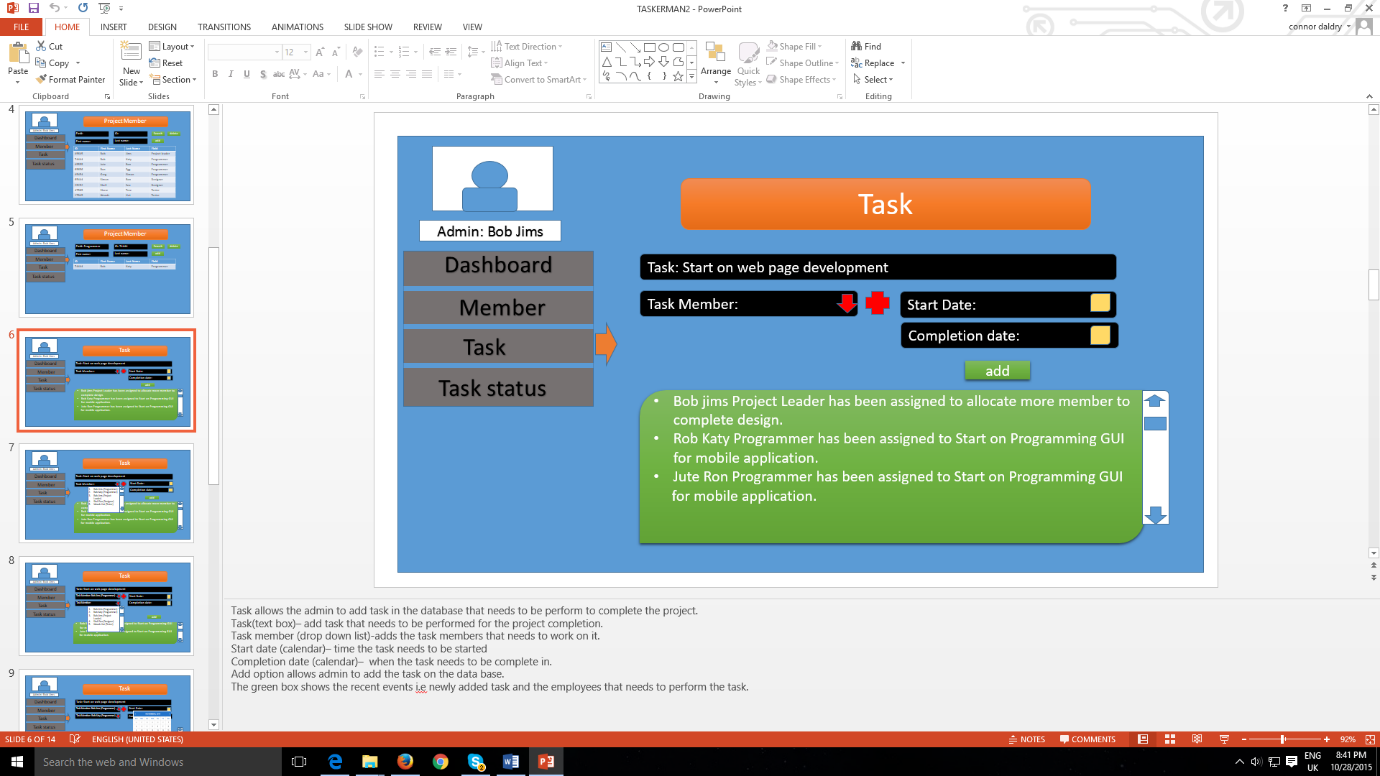
After user has fill all the field then the admin can use the add button to add the member in the database.



To delete employee the admin can use the field, ID, First name or Last name and use the search button to search the user.

It search through the database and shows the user.

Now admin can choose the name from result and use the delete option to delete the name from the database.



Task allows the admin to add task in the database that needs to be perform to complete the project.

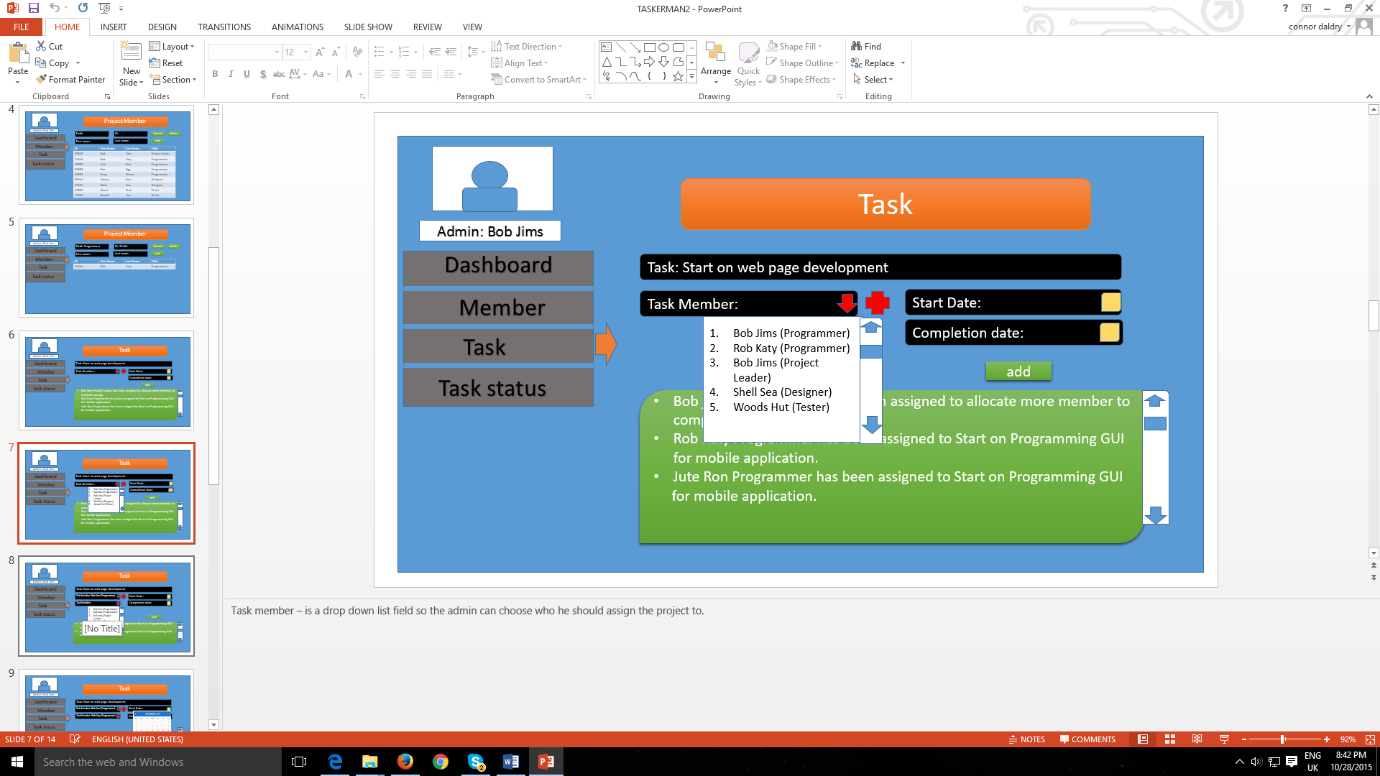
Task (text box) – add task that needs to be performed for the project completion.

Task member (drop down list)-adds the task members that needs to work on it.

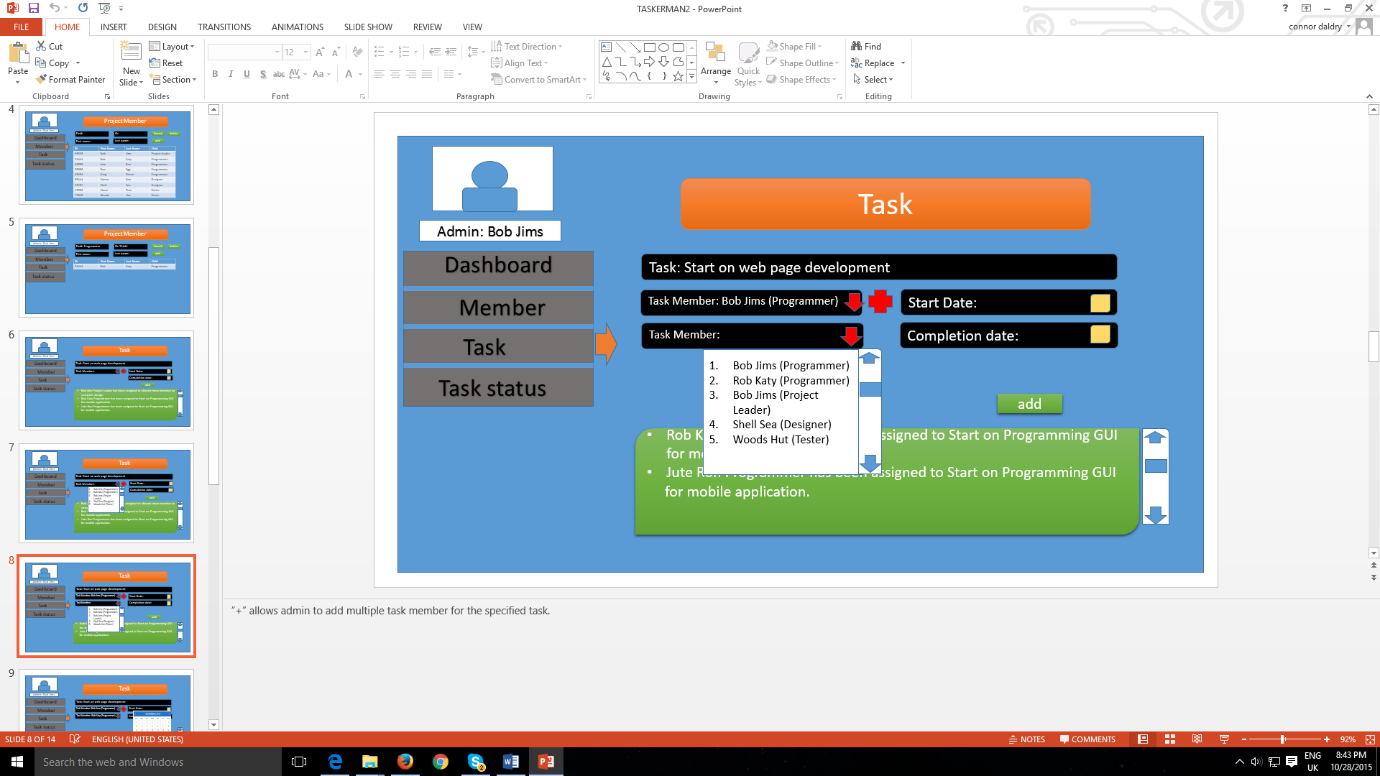
Start date (calendar) – time the task needs to be started

Completion date (calendar) – when the task needs to be complete in.

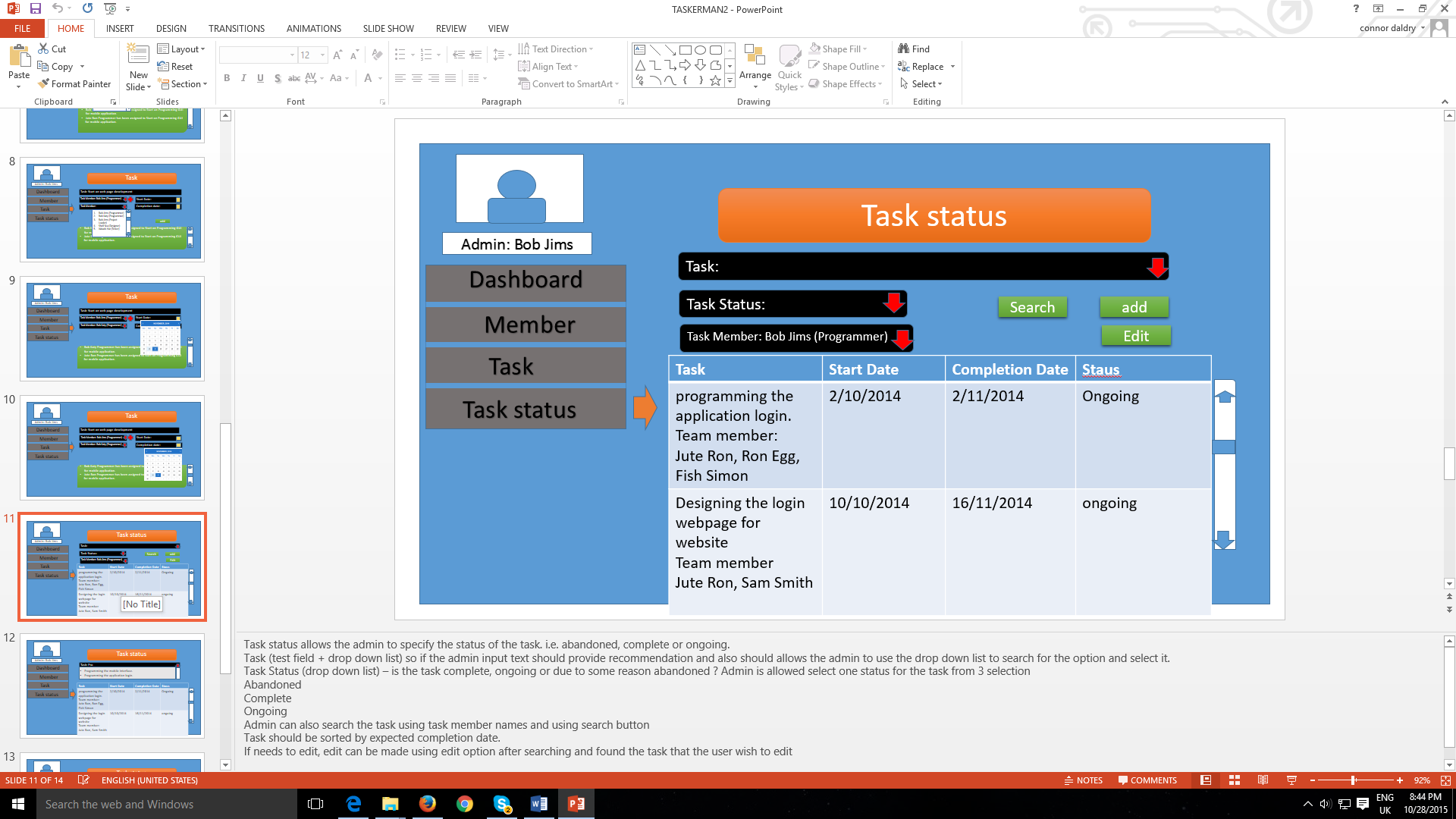
Add option allows admin to add the task on the data base.

The green box shows the recent events i.e. newly added task and the employees that needs to perform the task.

Task member – is a drop down list field so the admin can choose who he should assign the project to.



“+” allows admin to add multiple task member for the specified task.



Task status allows the admin to specify the status of the task. I.e. abandoned, complete or ongoing.

Task (test field + drop down list) so if the admin input text should provide recommendation and also should allow the admin to use the drop down list to search for the option and select it.

Task Status (drop down list) – is the task complete, ongoing or due to some reason abandoned? Admin is allowed select one status for the task from 3 selection

Abandoned

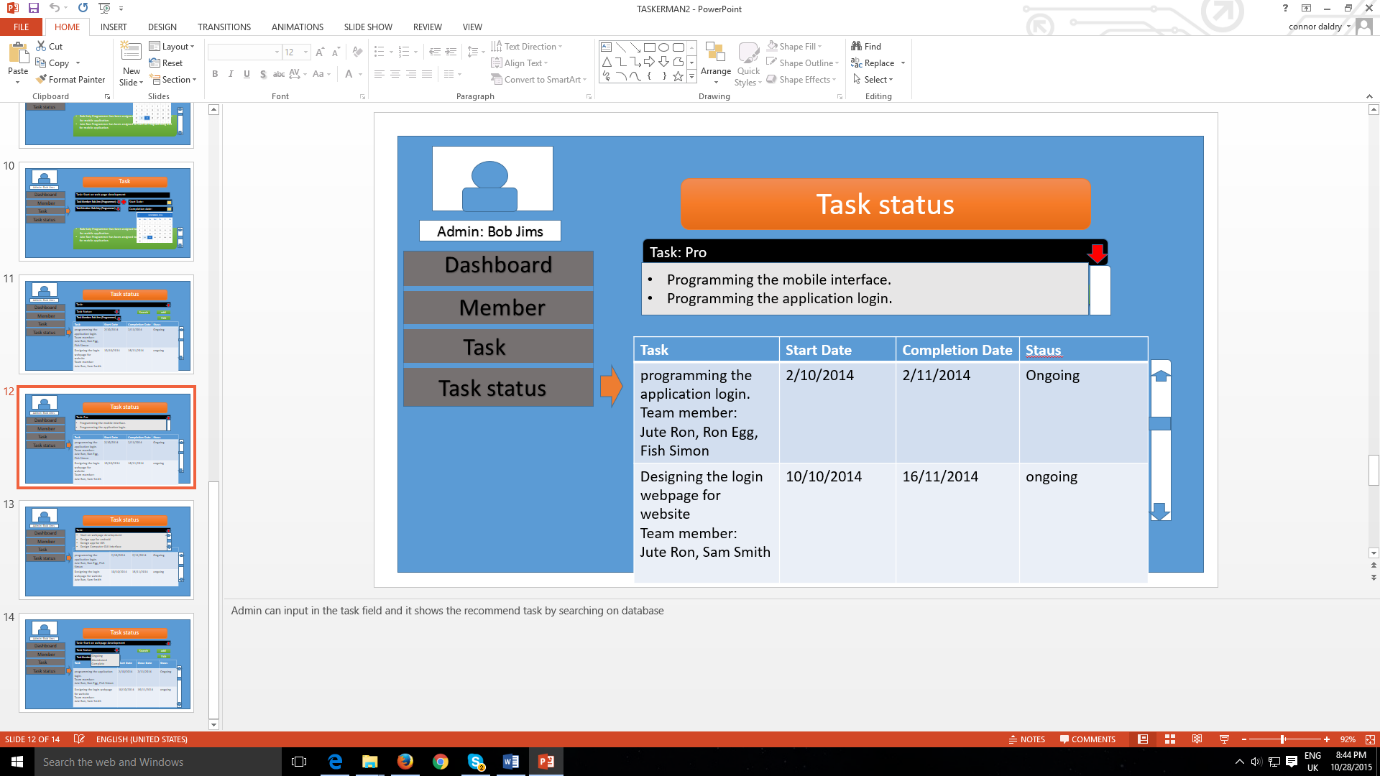
Complete

Ongoing

Admin can also search the task using task member names and using search button

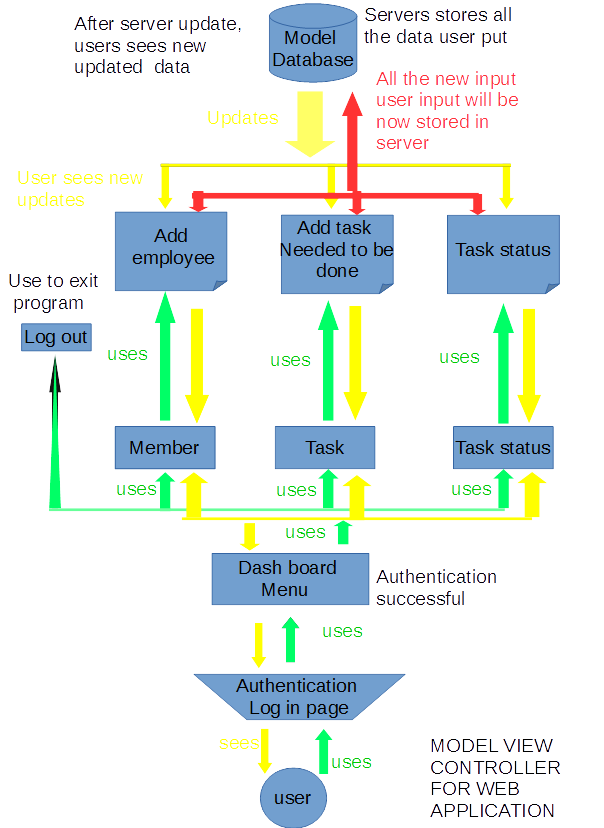
Task should be sorted by expected completion date.

If needs to edit, edit can be made using edit option after searching and found the task that the user wish to edit



Admin can input in the task field and it shows the recommend task by searching on database or the admin can use the drop down list to search the whole task added on the database, admin can use the drop down list to choose one of the status option for the specified task.

# COMPONENT DESCRIPTION



The diagram shows how the user is going to work and response with the web application and vice versa. This diagram shows the layer of things that the user needs to proceeds to starts the application, i.e. by providing the valid email id and password and after the user successful authentication user is allowed to enter the web application. After user get access to the web application the diagram shows that the user will be sent to the dashboard web page which provides the menu to navigate themselves and utilize the function provided properly. The diagram shows various web pages that are shown as a controller to manipulate the data in the database. These are basically the web pages providing users to store the employees in the database create a task and its details and store it and such. After user manipulates the data using the function provided the server updates the data and provides user the latest data which user can see/view. After the user has completed the task he/she can use the logout option to safely exit the web application.

# SIGNIFICANT CLASSES

The significant classes are definitely useful in our TaskerCLI as they determine on how the software will be structured upon the classes given below. They are an important aspect in order to develop the software further as well as keeping the backbone of the interface so that the program is much easier to handle from the developer point of view.

## – Interface Description

**Login()**

Extends javax.swing.JFrame

Public Methods – public Login(), public static void main(String args[]), public void run()

This class is for the user to log in to the application.

**Menu()**

Extends javax.swing.JFrame

Public Methods – public Menu(), public static void main(String args[]), public void run()

This class controls the menu within the application

**EmployeeList()**

Extends javax.swing.JFrame.

Public Methods - public static void main(String args[]), public EmployeeList(), public void run()

Holds the list of employees.

**TaskList()**

Extends javax.swing.JFrame

Public Methods \_ public TaskList(),public static void main(String args[]), public void run()

Holds the list of tasks. Can add tasks to the list and checks the status of the tasks.

**AllocateTasks()**

Extends javax.swing.JFrame

Public Methods – public AllocateTasks(), public static void main(String args[]), public void run()

Assigns a tasks to one of the employees.

**MonitorTasks()**

Extends javax.swing.JFrame

Public Methods – public MonitorTasks(), public static void main(String args[]), public void run()

Checks the status of the tasks and how long until they are due to be finished.

**UseMainFrame()**

Extends javax.swing.JFrame

Public Methods – public UserMainFrame(), public static void main(String args[]), public void run()

This class is used to navigate through the CLI.

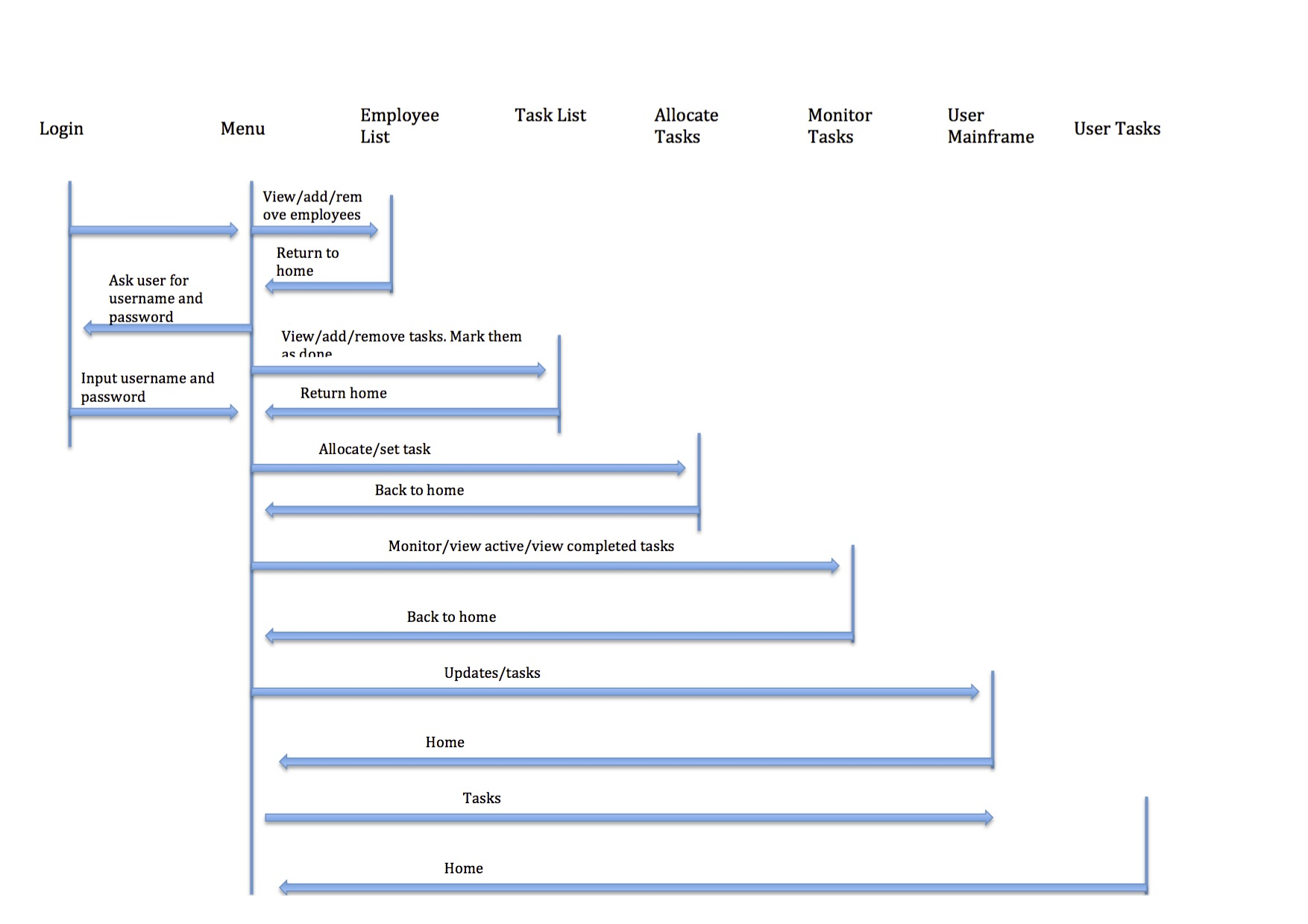
**UserTasks()**

Extends javax.swing.JFrame

Public Methods – public UserTasks(), public static void main(String args[]), public void run()

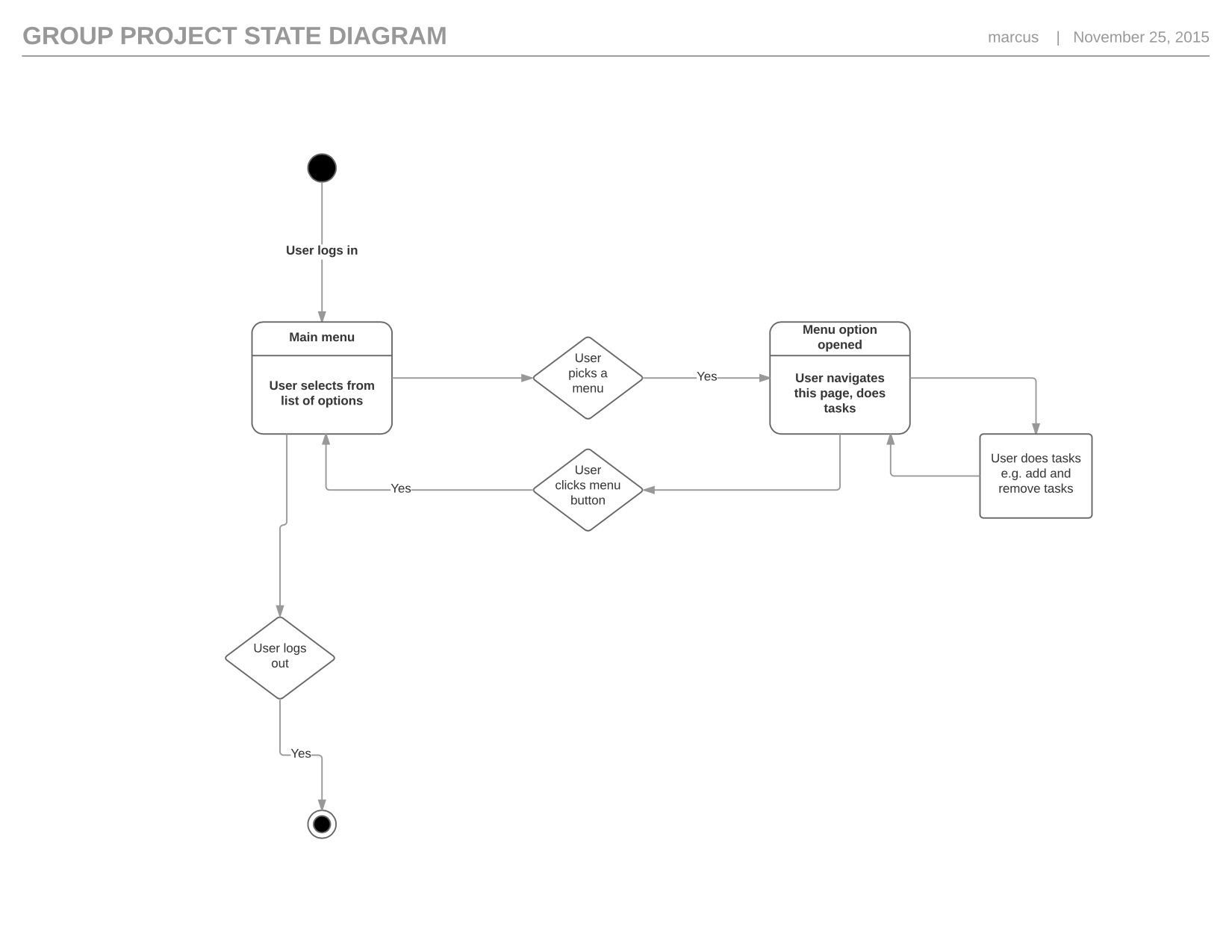
This class shows the user their own tasks with all of the information about the tasks.

# DETAILED DESIGN

**5.1 – UML Sequence Diagram**

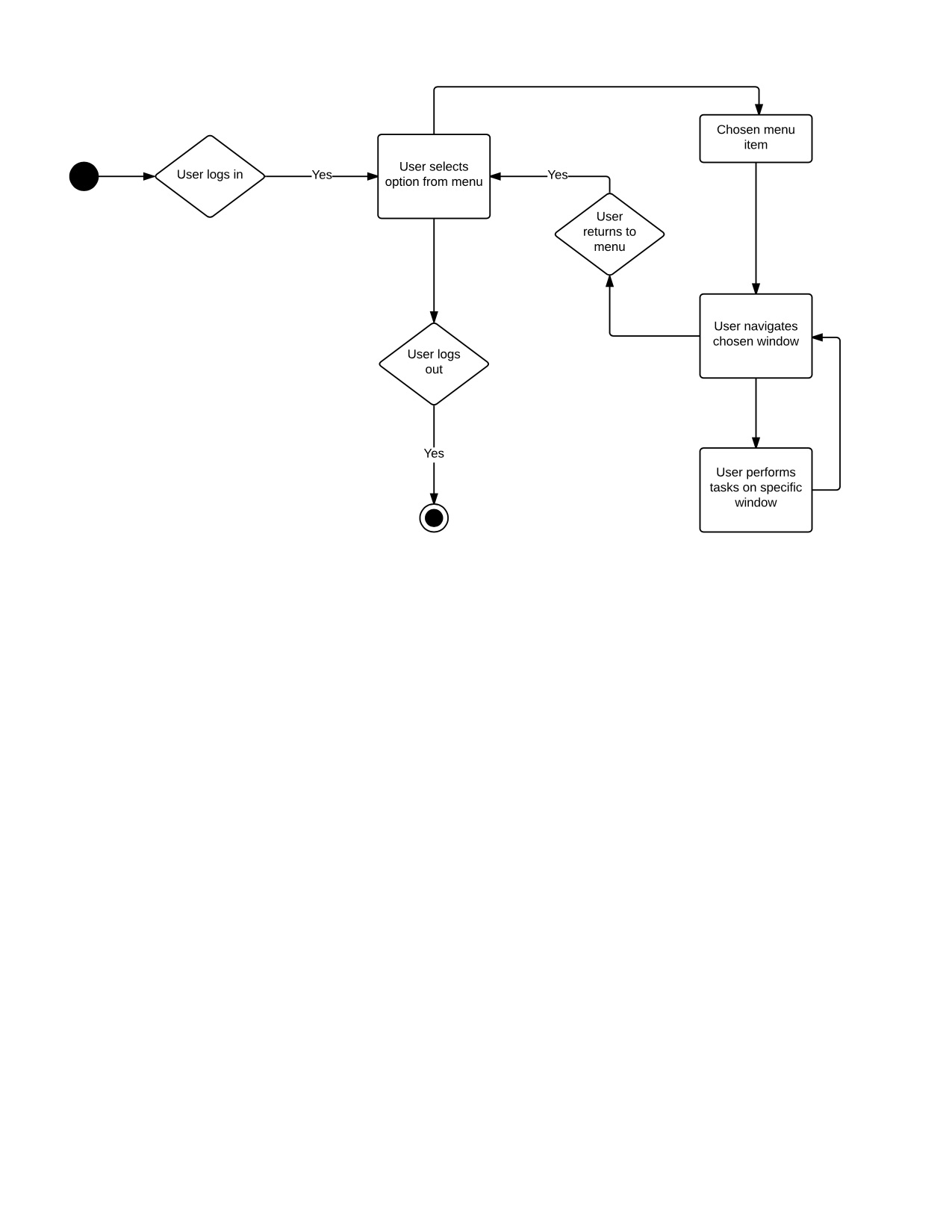
The UML Sequence diagram there describes the processes that are carried out once the client uses the software, which will be created. As you can see everything begins at the login stage where the user will be asked for input of their own name in order to be able to access the menu. The menu therefore is linked to every other processes within our web and desktop clients as the user will have to be able to navigate smoothly so that the program is reliable, easy to navigate through and aesthetic. Therefore from the diagram you can also see how each of the processes links back to the menu, emphasising the flexibility of both TaskerMAN and TaskerCLI.

**5.2 – State Diagram**

****

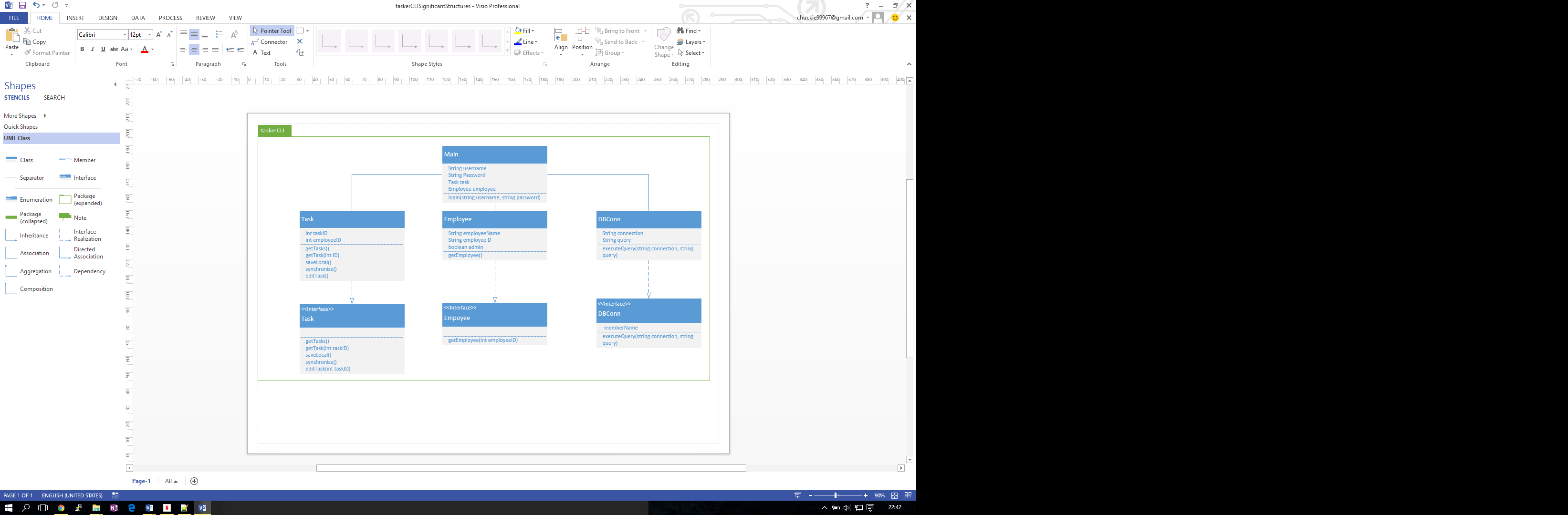
As you can see the state diagram describes the main functions that are going to be present in the Tasker from the client side of view. The state diagram therefore clearly shows the processes of a client using the software in order to navigate through the web or desktop clients from the point of logging in to the software and also logging out of it. In both web client and the desktop program a user will navigate through a menu in order to add and/or remove tasks from the database, which both pieces software will connect to and store the data.

**5.3 – Activity Diagram**

****

The activity diagram, which is shown above, clearly describes the activity of both TaskerMAN and TaskerCLI, as they both will have the same purposes of which are to allocate tasks to its users. Just like the state diagram, the activity diagram begins at the login stage and ends at the logout section where in between the user is able to go through different options. The activity diagram also links very closely to the sequence diagram proving the flexibility, as again everything links back to the main menu, enabling the user to easily navigate through the both web and desktop clients.

**5.4 – Significant Data Structures**



The component structure for the TaskerCLI shown above shows the different methods, and attributes of each class. It’s highly important to outline such in the design specification to make it clear what should be implemented, clearly outlining what the system needs to accomplish. There will be three classes for the TaskerCLI, as well as the main, the main will provide the link between the classes, calling the classes function as necessary as the user controls the GUI, which will also be provided through the main file.

REFERENCES

* Requirements Specification – USed throughout document
* QUality Assurance Document (SE.QA.5A) – Used to ensure that the document is correct and of good quality
* QUALITY ASSURANCE DOCUMENT (SE.QA.3) – Used to make sure document is correct and of good quality
* <http://www.essentialstrategies.com/publications/modeling/uml.htm> (UML LANGUAGE DIAGRAM) – USed to gather information about UML Language Diagram’s
* <http://www.visual-paradigm.com/solution/sequencediagram/> (UML SEQUENCE DIAGRAM) – Used to gather information on UML Sequence Diagrams
* <https://en.m.wikipedia.org/wiki/Model–view–controller> (Model view Controller - USed to gather information on Model View Controllers

DOCUMENT HISTORY

| *Version* | *CCF No.* | *Date* | | *Changes made to document* | | | *Changed by* |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1.0 | N/A | 28/10/15 | | First Version | | | C.Palmer |
| 1.1 | N/A | 28/10/15 | | Added Designs for TaskerMAN | | C.Daldry | |
| 1.2 | N.A | 28/10/15 | | | Tidying up of Document | | D.Kastelik |
| 1.3 | N/A | 20/11/15 | | | Component Description | | K.Rai |
| 1.4 | N/A | 22/11/15 | | | Significant Classes | | S.Johanssen |
| 1.5 | N/A | 24/11/15 | | | Detailed Design | | M.Hill |
| 1.6 | N/A | 24/11/15 | | | Changes to structure and layout | | D.Kastelik |
| 1.7 | N/A | 25/11/15 | | | Overview of document | | All Members |
| 1.8 | N/A | | 25/01/2016 | | Made changes to References to provide more detail | | C Griffiths |
| 1.9 | N/A | | 26/01/2016 | | Kimit modified Component Description and Curtis reviewed it and added it in. | | K Rai, C Griffiths |