Software Engineering Group Project
End of Project Report

Author:	Ben Dudley, David Fairbrother, Jonathan Englund, Josh Doyle, Liam Fitzgerald, Maurice Corriette, Oliver Earl, Tim Anderson
Config Ref:	SE_05_DEL_06
Date:	2016-13-02
Version:	1.0
Status:	Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document contains the teams learning outcomes, critical feedback and performance reviews. It additionally specifies improvements for future projects [1].

1.2 Scope

This report describes the final state of the project and outcomes following the completion of the project.

1.3 Objectives

The reader of this document should understand new mitigation techniques for the difficulties encountered during the completion of the project. It should additionally provide new processes for software developments in future projects.

2. MANAGEMENT SUMMARY

We set out to create a task assignment system with three components, TaskerCLI, TaskerMAN and TaskerSRV. These components communicate to each other via TaskerSRV and allow for task assignment and synchronisation. The client provided a description of each component and their functional requirements [2] to ensure each component meets their expectations.
All members of the team were open and friendly from the beginning. This meant everyone quickly learnt individual strengths and weaknesses which allowed us to split into two separate teams; web and client development. Every member of the team provided strong contributions to the project and was happy to receive constructive criticism and incorporate changes leading to a strong feeling of comradery.

The biggest difficulty continuously encountered throughout this project was lack of experience or exposure to elements used within this project. This included building GUIs, using JDBC in Java, displaying modal windows and handling a variable number of elements in the management component. Prototyping provided some initial experience and lead to subsequent design changes. During the projects implementation bugs arose due to the team not fully understanding the technologies; this led to subsequent refactoring of earlier code as more principles were understood.

Within TaskerCLI race conditions and threading lead to unexpected and difficult to reproduce bugs. This ranged from the database disconnecting randomly to all windows of the program becoming invisible. The client development team developed a stronger understanding of how to use the debugger to reproduce these race conditions and prevent them.
The team developing TaskerMAN was struggling to post data to modal windows. Initially they created several prototypes using AJAX however they later realised that through the use of PHP GET and POST methods they could easily push data to the page. This had the additional benefit of sanitising data when combined with prepared statements.

At the conclusion of this project out of the twelve functional requirements we completed and demonstrated all but two functional requirements. The two functional requirements not completed are FR7, filtering and sorting of tasks in TaskerMAN and FR9, task synchronisation in TaskerCLI.
FR7 was not implemented as the requirement was misread as being optional. FR9 is implemented however two bugs prevent the correct operation of saving elements and updating a task on reconnection.
All documentation is currently in a good state as they contain the latest information on the project and meet QA standards.

3. HISTORICAL ACCOUNT

Once the initial functional requirements had been provided for the Tasker system, the group set about deciding upon the platform that the Tasker system would be built on and tested against. They then produced a high level design which included user interface mock ups and UML diagrams to "map out" the overall functionality. This high level design was shown to the client to ensure that the group effort was being applied in the correct way. From that meeting with the client the group learnt that our understanding of Task Elements was incorrect; this delayed development until the group figured out the best way to implement the changes in the user interface design, and the design of TaskerCLI, TaskerMAN, and TaskerSRV.

Once initial designs had been agreed upon, the group was separated into two sub-groups to separately work on development of TaskerCLI and TaskerMAN. This was done to allow the group member's skills to be applied more effectively in their strongest area.

3.1 TaskerCLI

The first task for the team handling TaskerCLI was to begin work on the Design Specification whilst the Test Specification was being developed.

The Design Specification contains some contents from the high level design document, which had already been completed. However, due to changes stemming from the production of the high level design documentation, these diagrams could not be copied over without modification. We decided that the team working on TaskerCLI would not contribute heavily to the Test Specification and only work on the Design Specification for TaskerCLI. This was done to ensure that all changes could be made before the Design Specification deadline arrived.

This change to how Task Elements were handled, as pairs of descriptions and comments as opposed to a single element, required changes to the user interface design as well as a lot of the logic of loading and storing data. Interfaces and Class Diagrams also had be developed in accordance with this new way of handling Task Elements.

After the Design Specification had been submitted, work began on building the prototype of TaskerCLI for demonstration to the client. The major difficulty in developing this prototype was that TaskerSRV was not functional during development, meaning the desired functionality (downloading Task data) could not be implemented in time for the prototype demonstration. Fortunately, however, TaskerCLI had the ability to work in an "offline mode" so the basic functionality could be displayed.

Development of the final software was a generally straightforward task, however there were a series of bugs found during testing that extended the development time. The most difficult of the errors to find and fix were race conditions that arose when working with TaskerSRV. These errors were arduous and took many hours to fix, meaning that some team members were required to work extra hours in order to complete the development of the software.

3.2 TaskerMAN

The first task for the PHP team was the design specification in joint progress with the TaskerCLI team whom were also working on the design specification. The creation of the design specification initially when on without much of an issue, however there were timetabling conflicts which resulted in TaskerMAN team meetings without all members present.

After the submission of the test specification, the feedback introduced a curveball which affected the design stage, as we were unaware of the complexity behind task elements. We believed that Task elements was nothing more than a long description of the task.

During the prototyping stage of the project the entire team ran into issues due to TaskerSRV not being functional which held back all other development. Thankfully TaskerMAN was somewhat functional as we had modal windows and login working correctly.

When it came to actually implementing the PHP the hardest issue to deal with was finding a creative solution for inserting a user defined number of task elements which was eventually tackled but not without its struggles.
Aside from this difficulty the implementation was relatively calm without any major hiccups or setbacks. Bugs were identified and patched as coding week progressed until a fully operational website was up. All functional requirements were met aside from the filtering of tasks. This was a functional requirement that was unfortunately completely forgotten during implementation

3.3 TaskerSRV

TaskerSRV was planned with the functional requirements in mind to contain three tables: Users, Tasks and Elements. Whilst the users and tasks table design did not change throughout the project a misunderstanding about foreign keys led to the elements table being redesigned.
Previously the elements table contained two foreign keys and a string to allow elements to map to a user and task. After discussing with the client the team realised we needed to store a comment and the current design would not work.
This lead to us redesigning the elements table to hold a primary index, comment and element description and a foreign key to the relevant task.
The next step was creating this schema from the design, for this task we used MYSQL workbench which allowed us to graphically prototype the database. After some minor changes such as the length of a column being too short we verified TaskerSRV's operation.
Using the queries generated from MYSQL we created a BASH script to generate a fresh database. This script can be run interactively or from a single command and will detect an existing installation, or partial installation and generate clean tables with the user's permission. The final step was taking test data produced and using the bash script to automatically insert this data when instructed to create test data by the user.

4. FINAL STATE OF THE PROJECT

The project is viewed as meeting all but two functional requirements at its completion.
TaskerSRV correctly implements all functional requirements; this includes storing the following information within its schema:

- Full name and Email Address for Users
- Task Title, Start Date, End Date, Status and Allocation
- Task Element with free text comments

TaskerMAN meets all but one functional requirements; these are:

- Adding, Updating and Deleting team members
- Adding, Updating, Viewing and Deleting task data
- Reallocation of tasks
- Identification of managers

This includes input validation and additional checking such as enforcing a member cannot have any tasks before being deleted. Filtering and sorting by column specified in FR7 has not been implemented so FR7 is not completely met.

TaskerCLI implements all but two functional requirements:

- Connecting to database and storage of connection details
- Allowing users to login and local storage of this list
- Local copies of tasks periodically updated from TaskerSRV (every minute currently)
- Intuitive GUI (IR1)

Due to a bug in saving task elements an updated comment will not synchronise back to the database. This same code path also is responsible for uploading changes made offline as soon as a connection is available thus offline changes are discarded. Despite this changing a Task's status still synchronises correctly whilst online.
Secondly whilst offline the database will automatically attempt to reconnect on a timer, this leads to multiple dialog boxes being displayed every time a connection is attempted if the application was previously online. This can make the program difficult to use due to dialog boxes appearing on every change or periodically.

5. LOCATION OF GROUP REPOSITORY

The group repository is located at:

https://github.com/antibones/CS221_Group_05

The final branch for delivery is the "master" branch.

6. PERFORMANCE OF EACH MEMBER

6.1 Ben Dudley

Ben worked mainly with the TaskerCLI and TaskerSRV team throughout the group project.
After an unsure start to the project, Ben was able to show his strengths within the group and proved to be an important member of the team. Initially he had limited experience with many of the aspects that were involved. However, he managed to learn many tasks with assistance from the group including Junit Testing, Git commands, UML mark-up using Visual Paradigm and integration testing which was completely new to him.
Ben was keen to gain experience in new areas; an example of this is when he prototyped a script to automatically create a TaskerSRV instance. The team had no previous experience in in this area, his experience gained whilst implementing the prototype helped create the final scripts for setting up the database.
Ben showed hard work and dedication throughout and was always keen to help other members of the group wherever possible.

6.2 David Fairbrother

David took the role of project leader and a java programmer in this project.
To effectively lead he organised the project into assignable tasks and created deadlines to ensure delivery times were met. If problems or ambiguity occurred he communicated with the client or team members to resolve those issues. He additionally created two development teams, this way team leaders can get a stronger idea of their member's strengths and weaknesses and decide the best member to assign a task to.
David also participated as a Java programmer. During this project he researched and implemented a Database class. This utilised JDBC to connect to TaskerSRV to perform task synchronisation. He then additionally researched debugging threaded applications and taught other Java programmers more advanced debugging.
David has been a strong team member both managing the project whilst providing support and valuable contributions to other individuals within the team.

6.3 Jonathan Englund

At the start of the project Jonathan was assigned to be a part of the Java team. Despite him having more experience in web development he wanted to gain experience in Java by completing spike work as well as debugging code that the others had created. He additionally created several diagrams and mock ups used within multiple design deliverables. During development Jonathan used his new experience to create several methods which were essential for example email validation on the login screen.
After the first day of development he helped the Web team as they required more people to complete additional work for the final deadline. Whilst working with the web team he developed the taskAdd function, to integrate with TaskerSRV he developed a function for converting input from the user into SQL queries. Although Jonathan had less experience than some others in the team he did provided strong valuable contributions and was willing to accept feedback.
Jonathan has learnt many things during this project and looks forwards to his next project where he will be able to contribute more as a result of these new skills.

6.4 Joshua Doyle

Before the development of the software started, Josh worked very closely on the development of the design of TaskerCLI, as he was one of the most experienced members in using Java. Josh designed the user interface for TaskerCLI and worked with David Fairbrother to design the Class structure of the TaskerCLI program.

When it came to development Josh was the head of the team working on TaskerCLI. In taking on this role, Josh was in charge of distributing tasks to everyone working on TaskerCLI. Josh also worked on writing code directly due to his long term experience of programming. Josh used Window Builder in Eclipse to create the GUI windows for the entirety of TaskerCLI, and wrote the code that saved and loaded Task and Member data. He performed extensive debugging and resolved many complex bugs discovered during the testing phase.

Josh's code was well-written however the code required to load Task Elements was quite complicated and was not commented sufficiently. This lead to him subsequently refactoring the code resolving several bugs discovered during testing.

6.5 Liam Fitzgerald

Throughout the project Liam attended all meetings and provided valuable contributions within them despite being initially shy. One of his first tasks he completed was UI spike work for TaskerMAN, the initial ideas from this spike work still form the foundations for our final website design.
Liam also worked with the TaskerMAN team and produced several documents for the design deliverable such as sequence diagrams for TaskerMAN. He later produced the basic HTML and laid the CSS foundations.
Unfortunately due to personal problems Liam was not able to work from implementation week onwards. After contacting the project leader to inform him of the situation early in the week the team reorganised the tasks allowing the project to still be completed. This was only possible due to him informing as early he did.

6.6 Maurice Corriette

From the beginning of the group project Maurice was determined to work hard in order to support the group and achieve the best grades possible. From the outset he has been committed, attending all group meetings and formal meetings whilst providing strong input in all.

From the start of the project Maurice identified his experience in testing and thus was selected as the group testing lead. As the testing lead Maurice took on the sole responsibility of the testing specification and testing report, with input from Oliver and David in QA sessions. All versions of the testing specification have been detailed and extensive enabling delivery of a high quality product which conforms to documentation. Furthermore due to Maurice's meticulousness in creating the test specification based on the requirements specification he noticed the design deviated from the requirements. This allowed him to alert the group of this deviation and his input was valuable in checking future design documentation conformed to the client's requirements.
During Testing and coding week Maurice worked with the developers. He tested the system extensively, opening issues on GitHub, walking the team through reproduction of those bugs, and ensuring the system adhered to the functional requirements.

6.7 Oliver Earl

Oliver undertook the roles of lead QA as well as the lead web developer in this group project. As the most experienced PHP programmer in the group, the development and bulk of the implementation of the TaskerMAN component fell under his jurisdiction. This includes the underlying logic that generates database queries and the presentational layer that the user navigates. He also worked closely with members in the group responsible for testing to ensure that the HTML5 and JavaScript validation was fit for purpose.

Oliver also ensured other members of the Web Team were fully involved wherever possible with tasks such such as presentational work to ensure everyone had work to complete.

Oliver carried out his QA duties in meetings, working alongside the Deputy QA and Team Leader to ensure documentation and all deliverables were in accordance with the functional requirements. Their thorough, systematic approach resulted in the achievement of consistent, positive feedback.

Punctual, hard-working and sometimes a bit of a perfectionist, Oliver ensured that he attended as many necessary meetings and group work sessions as possible to ensure work was done at a consistently high quality.

6.8 Tim Anderson

Tim took the role of deputy leader during this project. Tim fulfilled the duties in this role at the start of the project when the team leader wasn't present. He picked up the partially completed tasks and communicated the projects status to the client during the meeting ensuring work could continue.
Tim proved to be hardworking and a good member of the team who was able to communicate well with all. As a part of the web team he showed a good understanding of CSS, being responsible for the complete overhaul of the old design and crafting the new one individually. He helped with debugging and developing new HTML for TaskerMAN. This ensured the layout and presentation of the finished product matched the design requirements.
Furthermore he completed additional research work independently and used that knowledge to aid the advancement of the project. For example he researched PHPUnit however we did not manage to implement them.

7. CRITICAL EVALUATION

The team performed very well during this project even during setbacks faced throughout the project. All members clearly knew their areas of expertise and weakness allowing the team to best utilise individual skills. An area for improvement is during quieter weeks additional training could have been performed ensuring skill sets are more equally spread across team members. During this project all members were present throughout however if somebody with a unique essential skillset went ill it would have caused delays.

The project could have been improved in two major ways. Firstly the final product did not have any unit tests built for it. Initially we were planning on using test driven development; however our lack of experience with the technology meant whilst tests would pass the program would not function correctly.
Following this the team agreed additional prototyping would have paid dividends. Our initial prototypes proved concepts such as using window builders to create GUIs but did not complete additional steps such as refreshing the displayed data. This lead to large deviations from the original plan which assumed certain functionalities that could have been avoided with more rigorous prototyping.
From this extra prototyping new unit tests should be built, a common issue during implementation was regressions which went largely undetected from the lack of unit tests. Whilst unit tests would have taken additional time to produce, the time saved would have paid dividends.

The team developed an understanding of the need for a well-developed project plan. By completing research and identifying risks in new technology additional testing and training can take place. This prevents large design changes, unnecessary refactoring and bug introduction. In larger projects with multiple teams these changes would significantly impact on other teams who may depend on the old incorrect design.
Finally within more complex methods it is worth breaking them into sequence diagrams. Whilst the implementation may seem trivial and everyone will still approach that algorithm differently. This leads to people assuming the algorithms behaves in a certain manner as it is not documented and leads to subsequent bugs stemming from those assumptions.

REFERENCES

[1] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.11 1.9 - Producing a Final Report,* Aberystwyth University: Software Engineering Group Project, 2016.

[2] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	13/02/2016	Original Version	DAF5

Software Engineering Group Project
Testing Report

Author: David Fairbrother, Maurice Corriette
Config Ref: SE_05_DEL_05
Date: 2016-02-14
Version: 1.0
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

INTRODUCTION

The tests in this document are based on Test Specification 2.1 [1]

This document contains a report [2] listing the results of the tests carried out on the Tasker system. The descriptions of the tests are in the test specification 2.1.

This report additionally states the tester, date of tests, CCF, and the version of the system tested.

This document should be used for guidance on the work various components still need completing should more time be allocated to development.

TEST LOG FORM

Test log No: 001
Group: 05
Testers: Maurice Corriette (Mac81) Ben Dudley (bed19)
Date: 12/02/15

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS1	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS2	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS3	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS4	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS5	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS6	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS7	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS8	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS9	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS10	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS11	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS12	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS13	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS14	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS15	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS16	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS17	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS18	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS19	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS20	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS21	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS22	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS23	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS24	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS25	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS26	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS27	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS28	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS29	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS30	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS31	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS32	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS33	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS34	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS35	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS36	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS37	FAIL	TaskerMAN allows for two characters, rather than three as a minimum, and no error message is provided.	#179	Maurice Corriette	28/01/16	1.0
TS38	Fail	The end date is accepted regardless of it being before the start date and no error message is provided.	#191	Maurice Corriette	28/01/16	1.0
TS39	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS40	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS41	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS42	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS43	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS44	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS45	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS46	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS47	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS48	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS49	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS50	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS51	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS52		N/A	N/A	Maurice Corriette	28/01/16	1.0
TS53	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS54	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS55	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS56	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS57	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS58	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS59	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS60	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS61	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS62	FAIL	TaskerMAN allows tasks with the same name, this is an intentional change in our design and the finished software reflects this change	#167	Maurice Corriette	28/01/16	1.0
TS63	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS64	FAIL	Letters are accepted as input and this causes a fatal allowed memory error.	#190	Maurice Corriette	29/01/16	1.0
TS65	FAIL	Special characters are allowed as input and this causes a fatal memory error	#179	Maurice Corriette	29/01/16	1.0
TS66	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS67	Pass	N/A	N/A	Maurice Corriette Maurice Corriette	29/01/16	1.0
TS68	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS69	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS70	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS71	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS72	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS73	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS74	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS75	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS76	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS77	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS78	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS79	FAIL	The end date is accepted regardless of it being before the start date and no error message is provided.	#191	Maurice Corriette	29/01/16	1.0
TS80	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS81	FAIL	The start date is accepted regardless of it being after the end date and no error message is provided.	#191	Maurice Corriette	29/01/16	1.0
TS82	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS83	FAIL	TaskerMAN allows tasks with the same name, this is an intentional change in our design and the finished software reflects this change.	#167	Maurice Corriette	29/01/16	1.0
TS84	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS85	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS86	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS87	Pass	N/A	N/A	Ben Dudley	29/01/16	1.0
TS88	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS89	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS90	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS91	FAIL	An error message is not provided informing user that task has no elements	#192	Ben Dudley	29/01/16	1.0
TS92	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS93	FAIL	An error message is not provided informing the user that the task cannot be edited, furthermore user is allowed to edit a task whether it has been abandoned or not. Tis should not be the case.	#195	Maurice Corriette	29/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS94	FAIL	User is allowed to delete a task regardless of whether it is still allocated or not, furthermore no error message is provided	#194	Maurice Corriette	29/01/16	1.0
TS95	Pass	N/A	N/A	Maurice Corriette	29/01/16	1.0
TS96	FAIL	Filtering by end date has not been implemented.	#196	Maurice Corriette	29/01/16	1.0
TS97	FAIL	Filtering by status has not been implemented.	#196	Maurice Corriette	29/01/16	1.0
TS98	Pass	N/A	N/A	Maurice Corriette	27/01/16	1.0
TS99	Pass	N/A	N/A	Maurice Corriette	27/01/16	1.0
TS100	Pass	N/A	N/A	Maurice Corriette	27/01/16	1.0
TS101	Pass	N/A	N/A	Maurice Corriette	27/01/16	1.0
TS102	FAIL	This was not implemented, as this was a requirement believed to be optional and it was not stated in the requirements specification for TaskerCLI	#197	Ben Dudley	27/01/16	1.0
TS103	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS104	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS105	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS106	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS107	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS108	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS109	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS110	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS111	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS112	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	27/01/16	1.0
TS113	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS114	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS115	Pass	Dynamic connection colour status intentionally not implemented	N/A	Ben Dudley	28/01/16	1.0
TS116	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0
TS117	Pass	N/A	N/A	Ben Dudley	27/01/16	1.0
TS118	Pass	N/A	N/A	Maurice Corriette	27/01/16	1.0
TS119	FAIL	This was not implemented because it was unanimously decided to be an unnecessary and redundant feature similar to the dynamic connection colour status	N/A	Maurice Corriette	27/01/16	1.0
TS120	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0
TS121	FAIL	User can edit task status to ‘abandoned’. There is no error message provided.	#180	Ben Dudley	28/01/16	1.0
TS122	FAIL	User can edit task element comments to blank and save it. There is no error messages provided	#200	Ben Dudley	28/01/16	1.0
TS123	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0
TS124	FAIL	User is allowed to enter Special characters into task elements. No error messages provided	#180	Ben Dudley	28/01/16	1.0
TS125	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0

Test ID	Pass / Fail	Pass / Fail description	Issue #	Tester	Date of test	VS of system
TS126	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS127	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0
TS128	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0
TS129	Pass	N/A	N/A	Maurice Corriette	28/01/16	1.0
TS130	Pass	N/A	N/A	Ben Dudley	28/01/16	1.0

REFERENCES

[1] M. C. O. E. David Fairbrother, *Testing Specification 2.1,* Aberystwyth University: Software Engineering Group Project, 2016.

[2] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.11 1.9 - Producing a Final Report,* Aberystwyth University: Software Engineering Group Project, 2016.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	2016-02-14	Original Version	DAF5

Software Engineering Group Project
Maintenance Manual

Author:	Ben Dudley, David Fairbrother, Jonathan Englund, Josh Doyle, Liam Fitzgerald, Maurice Corriette, Oliver Earl, Tim Anderson
Config Ref:	SE_05_MAINT_01
Date:	2016-02-14
Version:	1.0
Status:	Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document contains essential information for the maintenance of TaskerCLI, TaskerMAN and TaskerSRV [1].

1.2 Scope

This information in this document pertains to future development, maintenance, installation and testing of the Tasker system.

1.3 Objectives

The reader of this document should be able to identify the offending code for future bugs, or the best location to add new features. It also forewarns of any potential pitfalls for the unwary programmer and how to test functionality of updated components.

2. TASKERCLI MAINTENANCE MANUAL

2.1 Program Description

TaskerCLI is a desktop application which allows users receive tasks. It allows users to change the task status, have multiple elements per task and add or edit elements for that task. It works both offline and online and will automatically synchronise to the database.

2.2 Program Structure

TaskerCLI has two groups of classes, GUI classes such as MainWindow and LoginWindow and logic classes such as Database and Task.
GUI classes handle program start up, display and shutdown. Logic classes connect to the database and provide storage of data such as members or tasks.

2.3 Algorithms Used

TaskerCLI populates the main display from “taskSaveFile.txt” (described below). This allows it to display data both online and offline. To achieve this it parses the save file into the respective objects used within the program. In the event we cannot parse the file we discard the contents and ask the database to retrieve a new copy.

The database automatically synchronizes and attempts to reconnect using Timers, when fired these trigger methods to contact the database. When data is successfully retrieved from the database it is converted into a MemberList or TaskList. The TaskList is passed to MainWindow which handles saving and updating the display whilst MemberList is saved by the database class to a flat text file.

When a task is edited to ensure it does not discard the user's changes which are in progress automatic synchronization is paused. When the user closes the editing window it immediately resumes after saving their changes.

2.4 Main Data Areas

All Member Objects are held within a MemberList Object, and all Task Objects are held within a TaskList Object. These list Objects are essentially just ArrayLists that have been abstracted into their own class. The vital

methods used for an ArrayList, such as get and add, are still accessible through the methods associated with the MemberList and TaskList Classes.

Two enumerations are used in TaskerCLI. These two enumerations are used to represent the state of the connection between TaskerCLI and the TaskerSRV server, and the completion status of a single Task.

The database object holds a JDBC connection to the database which is open during the operation of the program. The details used to establish the connection are also stored within the database object to allow automatic reconnection. It also holds a reference to the main window which allows it to display error and warning messages.

2.5 Files

- **Save Files**
  - TaskerCLI uses four save files to save member and tasks data, TaskerSRV credentials to enable automatic logins and tasks pending synchronisation to the TaskerSRV database. These files need to be located in the same directory as the TaskerCLI .jar file.
  - **Member Save File**
    - memberSaveFile.txt
    - Holds the names and email addresses of all the Members that are to be granted access to TaskerCLI
    - If the file does not exist, it will be created on start-up and automatically populated when TaskerCLI is connected to TaskerSRV
    - If the contents of the files becomes corrupted, its contents will be overwritten with Member data downloaded from TaskerSRV
  - **Task Save File**
    - taskSaveFile.txt
    - Holds the data associated with all tasks stored in TaskerSRV
    - If the file does not exist, it will be created on start-up and automatically populated when TaskerCLI is connected to TaskerSRV
    - If the contents of the file becomes corrupted, its contents will be overwritten with Task data downloaded from TaskerSRV
  - **TaskerSRV Credentials Save File**
    - connSaveFile.txt
    - Holds the credentials of the TaskerSRV database
    - Credentials are saved when the user connects to the TaskerSRV database
    - If the user changes the credentials from within TaskerCLI, the save file will be updated automatically
  - **Pending Tasks File**
    - pendingSaveFile.txt
    - Saves Tasks that could not be synced with TaskerSRV so they can be synced when TaskerCLI is next connected to TaskerSRV
    - Pending Tasks are added to the file automatically and are removed when they are synchronised with TaskerSRV

2.6 Interfaces

The program uses a MYSQL JDBC driver to communicate with the database. As different JDBC drivers use different connection strings a MYSQL database must be used with this program. It can use a custom port for the MYSQL database but if one is not specified it uses the default port of 3306.

2.7 Future Improvement

The program's usability can be further improved by displaying clearer and concise error messages describing what went wrong and how the user could fix it. Currently the program throws a dialog box if an automatic reconnect fails, as the user cannot change this behaviour it should not throw a dialog box.
Additionally a colour indicator could also show the current state of the connection and many text entry boxes don't accept enter requiring the user to use their mouse.

2.8 Potential Pitfalls

Extra care must be taken editing methods which save, load, get and set a TaskList or MemberList. If the save format is not adjusted correctly the program will not display any tasks or allow login as it will keep discarding the file. All get and set methods are threaded, this can lead to unforeseen race conditions. Care needs to be taken if the order of these methods need to be serialised as they execute in their own thread and don't return to the caller on completion.

Some of the tables used to display data in TaskerCLI appear to allow editing of cells. This, however, is not the correct method of editing data and any data entered into these cells will not be saved locally or synchronise with the TaskerSRV database. When editing Task Comments, selecting a Task Element from the table will display the Task Element in the two text boxes above the table. It is here that the Task Element Comment should be changed. Clicking the Submit button will then temporarily submit the changes, and clicking the Save button will commit changes locally.

2.9 Limitations

TaskerCLI requires very little processing power it has been tested and used with a 200 MHz processor
The program recommends 128MB of RAM with 64MB being minimum. Below this amount excessive swapping will occur severely degrading performance.
10MB of disk space is recommended whilst 4MB is minimum for the application and associated storage files.
The target system must also have a Java Runtime Environment installed for the program to function.

2.10 Rebuilding and Testing

To rebuild the project two external libraries are required: Swing MIGLayout 4.0 and MYSQL JDBC driver. Once added to the classpath the program should build. The program functionality is tested using the testing specification [2]. This is due to bugs mostly arising from race conditions which JUnit cannot easily detect.

3. TASKERMAN MAINTENCE MANUAL

3.1 Program Description

TaskerMAN is an online application (a website) that gives managers complete control over the users and tasks stored on the TaskerSRV database. It allows the creation, modification and deletion of users and tasks, including each task's individual elements.

Unlike TaskerCLI, TaskerMAN is only accessible by managers and also requires an Internet connection.

3.2 Program Structure

TaskerMAN is broken up into three sections - interfaces, logic and supporting files. The interfaces consist of three main files, 'index.php', 'taskerman.php' and 'users.php' - with the exception of 'index.php' which is the program's login page, these files use PHP includes to include other PHP files containing UI elements, such as modal windows for adding users, etc.

Pages such as 'editTask.php' which provides the code functionality for editing a task or 'connect.php' which provides a persistent database connection are logic files. They contain little to no HTML and are usually the POST target for forms contained on modal windows.

Supporting files include form validation JavaScript, stylesheets and images that are used by the program.

TaskerMAN is written procedurally, and therefore does not have classes, but does have global functions contained within the 'init.php' file, which is referenced by all files in the application.

3.3 Algorithms Used

There are no advanced algorithms used in TaskerMAN - however one of the most interesting or noteworthy is that which generates input textboxes with unique IDs based on a user defined number of elements.

- Take the number the elements the user wishes to add
- Use that number to construct a for loop
- Print input boxes with IDs based on the current loop iteration

3.4 Main Data Areas

The main data storage area used by the program are the files generated by the PHP Session system. These are server-side files stored in the tmp directory. While this is potentially unsafe - this could easily be reconfigured to any directory, including a directory outside of publicly accessible web folders. It is simply configured this way to ensure compatibility with Information Services servers.

3.5 Files

3.5.1 Interfaces:

- footer.php
  - Contains the footer for the TaskerMAN interface
- header.php
  - Contains the header for the TaskerMAN interface, including the navigation.
- index.php
  - Login page - the first page that users land on when accessing TaskerMAN
- meta.php

    - Contains HTML metadata and any non-PHP includes that are accessed sitewide, such as the CSS stylesheet.
- taskAdd.php
    - Modal window - provides functionality so that the user can add tasks to TaskerSRV
- taskEdit.php
    - Modal window - provides functionality so that the user can edit an existing task specified in the TaskerMAN main interface
- taskerman.php
    - The main TaskerMAN file - the Task view of the program. The bulk of program functionality is here - it uses PHP includes to load modal windows and other interface elements.
- taskView.php
    - Modal window - displays in detail the selected task
- userAdd.php
    - Modal window - provides functionality so that the user can add users to TaskerSTV
- userEdit.php
    - Modal window - provides functionality so that the user can edit an existing user specified in the TaskerMAN user interface
- users.php
    - The Users view of the program
- addElements.php
    - Modal window - allows the user to add a predetermined amount of task elements and their comments to a task
- additionalElement.php
    - Modal window - allows the user to add an additional element to an existing task

3.5.2 Logic

- init.php
    - One of the most important files in TaskerMAN. This file provides access to global functions, such as those responsible for error handling, provides a site-wide database connection and is responsible for providing application persistence by configuring a PHP session. This file is called by almost every PHP file.
- taskAddGateway.php
    - Acts as a gateway between the Add Task and Add Element modal windows and provides important data sanitisation
- taskDelete.php
    - Deletes the selected task from TaskerSRV
- userDelete.php
    - Deletes the selected user from TaskerSRV
- addEntry.php
    - Adds a task and its respective task elements/comments to TaskerSRV
- addExtraElement.php
    - Adds an extra element to an existing task
- addUser.php
    - Adds a user to TaskerSRV
- config.php
    - Site-wide configuration file, read by init.php
- connect.php
    - Configures and establishes a database connection
- editTask.php
    - Modifies an existing task entry in TaskerSRV
- editUser.php
    - Modifies an existing user in TaskerSRV
- elementDelete.php
    - Deletes all elements from an existing task

3.5.3 Supporting Files

- style.css
    - Stylesheet
- entryValidation.js
    - Validation for the login / index.php
- validation.js
    - Validation procedures for all remaining forms

3.6 Interfaces

TaskerMAN uses the PDO_MYSQL driver (that which itself implements the PDO interface) to communicate with TaskerSRV. Custom ports can be specified.

3.7 Future Improvement

One of the key requirements was accidentally left out during initial development: the ability to filter data displayed in the TaskerMAN main interface and sort it based on column values, namely sorting tasks based on their status. While this was a genuine small oversight, realistically this would be easy and low-risk to implement into the next version of the software, and could be done in a realistic timeframe of less than a few days.

TaskerMAN would however benefit highly from the refactoring of code in the future as there are some significant smells present in the source code, namely code repetition. Designating functionality to specific functions and reducing the amount of files would make software maintenance easier for future development. It would also be feasible to refactor the code to fit the object-oriented programming paradigm that PHP fully supports.

The use of AJAX was discussed as a means of dynamically updating both the TaskerMAN interface and passing data between modal windows without the need to POST data manually to the server. While this was deemed unnecessary and would add an extra layer of complexity to the software, experienced developers could take this into consideration.

One final major improvement identified by the developers is the lack of proper responsive design. While TaskerMAN behaves as expected in all major browsers, including mobile browsers, (with some presentational hiccups present when running under Internet Explorer), it was not designed to adapt its presentation based on certain characteristics, such as the low resolutions naturally present on mobile devices. A redesigned stylesheet and HTML layout would look to incorporate these features for maximum compatibility and ease of use.

3.8 Potential Pitfalls

As the layout of the program is not always necessarily clear, those maintaining the software must be careful to ensure that they do not break any dependencies as these will cause problems that cascade throughout the software. Regression testing will be important as code is refactored.

3.9 Limitations

As the majority of TaskerMAN is server-side, the main limitations imposed include an active Internet connection, a managerial account and a relatively modern web browser. HTML5 support is a plus as it affects the presentation of modal windows and the use of HTML5 validation, but is not a requirement.

As TaskerMAN is web based, it does not behave differently based on the user's operating system. Mobile devices and low resolutions are for the most part supported, but a minimum of 800x600 is generally recommended.

While the JavaScript and HTML5 was vigorously tested to ensure no invalid input could be sent to the program and that there is input sanitisation in use throughout the program to ensure no malicious input can make it to

TaskerSRV - it is by no means bulletproof. An experienced hacker or penetration tester would likely be able to bypass it easily, and of course, the system is more vulnerable if JavaScript is disabled, or if HTML5 is not supported by the user's browser.

3.10 Rebuilding and Testing

To deploy a new installation the requirements specified in the deployment description need to be met on the target server. Copy the tar containing TaskerMAN and automated install script and execute the script. This will setup the files, permissions and create the directory structure needed.

4. TASKERSRV MAINTENANCE MANUAL

4.1 Program Description

A MySQL database which holds tasks, users and their associated details. Both TaskerMAN and TaskerCLI synchronise to this database.

4.2 Tables and schema

TaskerSRV uses three tables to store data they are:

- tbl_users
  - Email (Primary Key, VARCHAR(45))
  - FirstName (Not NULL, VARCHAR(15))
  - LastName (Not NULL, VARCHAR(15))
  - IsManager (Not NULL, INT(11))
- tbl_tasks
  - TaskID (Primary Key, Not NULL, Auto Increment, Int(11))
  - TaskName (Not NULL, VARCHAR(45))
  - StartDate (Not NULL, DATE)
  - EndDate (Not NULL, DATE)
  - Status (Not NULL, INT(11))
  - TaskOwner (Foreign Key tbl_users->Email, Default NULL)
- tbl_elements
  - Index (Primary Key, Not NULL, Auto Increment, INT(11))
  - TaskDesc (Not NULL, VARCHAR(45))
  - TaskComments (VARCHAR(45), Default NULL)
  - TaskID (Foreign Key tbl_tasks->TaskID, Default NULL)

4.3 Rebuilding and Testing

In the event the database needs rebuilding the databaseSetup.sh script can be used. This can be used automated with command line flags or interactively. If a database already exists it will ask for confirmation before performing the operation and will reset the database to the original state. After resetting the login the only user accessible (unless test data is entered) will be “manager@example.com".
The script additionally will provide test users and test data if instructed to which can be used to test other applications such as TaskerMAN and TaskerCLI.

REFERENCES

[1] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.11 1.9 - Producing a Final Report,* Aberystwyth University: Software Engineering Group Project, 2016.

[2] M. C. O. E. David Fairbrother, *Testing Specification 2.1,* Aberystwyth University: Software Engineering Group Project, 2016.

[3] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	2016-02-14	Original Version	DAF5

Software Engineering Group Project
Project Plan

Author: David Fairbrother, Joshua Doyle
Config Ref: SE_05_PM_01
Date: 2015-10-29
Version: 1.2
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

This document outlines deadlines for deliverables and their requirements. It breaks these deliverables into tasks and creates a plan for their completion.

1.1 Purpose of this Document

The purpose of this document is to describe how tasks will be created, named and monitored to ensure deliverables are created within set deadlines and are of a high quality.

1.2 Scope

This document is written to assist in planning, timing and management of the project. It also lists tasks, deadlines and deliverables and creates a standard naming scheme and plan for their completion.

1.3 Objectives

This document aims to inform the reader of upcoming deliverables and their deadlines, it will break the deliverables down into assignable tasks with a standard naming convention. The project plan will also provide a schedule to detect problematic tasks and remediate the situation before time slippages occur. The reader should understand the timeline of the project and tasks required to complete it.

2. DELIVERABLES

2.1 List of deliverables and their deadlines

To ensure the project is progressing correctly and to a high standard, several deliverables will be presented at a review meeting with the project manager. These must be submitted via Blackboard before the meeting.

These deliverables and their deadlines [1] are:

1. Interaction and high level design for the system – October 30th
2. Test specification for the system – November 13th
3. Design specification for the final system – November 27th
4. 1st Prototype demonstration – December 11th
5. Delivery of software – January 29th by 16:00
6. Handover of all documentation – February 15th by 16:00

2.2 Additional important dates

There are other important dates within this project. These are listed below:

- Integration and testing week - January 25th – January 29th (Inclusive)
- Acceptance testing – Week commencing February 1st (times to be announced)

2.3 Individual deliverable requirements

2.3.1 Interaction and high level design for the system

Sections 4.1, 4.2, 5.1 and 5.2 within SE.QA.05A [2] are required for this deliverable. This includes:

- Applications in the system
- Application interactions
- Use-cases
- User interface design

2.3.2 Test specification for the final system

This deliverable requires the test specification to be completed as per SE.QA.06 [3] the documentation should include:

- Test specification

2.3.3 Design Specification

A full design specification must be created conforming to SE.QA.05A [2]. The following sections can be reused from Interaction and high level system design deliverable

- Applications in the system
- Application integrations
- Use-Cases
- User interfaces design

The following documentation must be created in addition:

- Component descriptions
- Significant classes
    - Interface design for these classes

To ensure a complex class or interaction is implemented correctly it also needs to be broken down into:

- Sequence diagrams
- State diagrams
- Activity diagrams
- Significant data structures

2.3.4 1st Prototype demonstration

A demonstration of the final product needs to be created; this is to allow the client to review and change elements of the product early in development. The demonstration must have all its major components completed however it does not need to be fully complete or polished.

2.3.5 Delivery of software

The software must contain all functional requirements and be polished and ready for customer delivery. The software will be delivered on a CD to the client. The client will then perform acceptance testing.

2.3.6 Handover of all documentation

All documentation must be completed and reviewed to ensure a consistent high standard across documents. All documents must also conform to the layout in SE.QA.03 [4]. Once finalised the documents will be exported to PDF and delivered to the client. The documentation includes [5]:

- Design specification
- Final report
- Maintenance manual
- Project plan
- Test report
- Test specification

3. TASK STANDARDS

3.1 Naming convention

All tasks will follow a standard naming scheme to allow identification of the deliverable targeted this is described in SE.QA.02 [6]. The format will be SE_05_XXXX_YY.

XXXX refers to the deliverable in a shorthand code such as PM for project management. YY is a numerical value starting and 01 and increasing for tasks which are large enough to warrant being broken down into subtasks. Tasks which are not suitable to be turned into sub tasks will have a subtask value of 01. [6]

3.2 Task groups

The following codes will be used for identifying tasks:

- DEGN – Design Specification
- MAINT – Maintenance Manual
- PM – Project Management
- QA – Quality Assurance
- TCLI – Tasker Client
- TEST– Testing Reports
- TEST_SPEC – Testing Specification
- TMAN – Tasker Manager
- TSRV – Tasker Server
- DEL – Complete Deliverable

For example a design document might have the code SE_05_DEGN_07 (if it was subtask 7)

4. LIST OF TASKS

4.1 Project Management Tasks

SE_05_PM_01 – Create Project plan documentation (This document)

SE_05_PM_02 – Create Gantt chart for project plan

Se_05_PM_03 – Create risk analysis for project plan

4.2 Test Specification Tasks

SE_05_TEST_SPEC_01 – Create test specification for TaskerCLI

SE_05_TEST_SPEC_02 – Create test specification for TaskerMAN

SE_05_TEST_SPEC_03 – Create test specification for TaskerSRV

SE_05_DEL_02 – Complete test specification deliverable

4.3 Design Specification Tasks

SE_05_DEGN_01 – Create system applications documentation

SE_05_DEGN_02 – Create application interaction documentation which lists to format data will be transmitted between software in

SE_05_DEGN_03 – Create use-case diagrams for the application

SE_05_DEGN_04 – Design and create mock-up of user interfaces

SE_05_DEL_01 – Interaction and high level design deliverable

SE_05_DEGN_05 – Document component level design

SE_05_DEGN_06 – Design and document significant classes within TaskerCLI

SE_05_DEGN_07 – Design and document significant functions and/or classes (where applicable) in TaskerMAN

SE_05_DEGN_08 – Design scheme and indexes for TaskerSRV database

SE_05_DEGN_09 – Break down complex designs and algorithms into UML sequence diagrams, state diagrams and activity diagrams where appropriate for TaskerCLI

SE_05_DEGN_10 – Break down complex designs and algorithms into UML sequence diagrams, state diagrams and activity diagrams where appropriate for TaskerMAN

SE_05_DEGN_11 – Spike work and document methods such as threading to conform to requirement PR1 [7] (all user interactions must take < 1 second to reflect their changes)

SE_05_DEGN_99 – Review all design documentation clarify and ambiguity and release

SE_05_DEL_03 – Complete design specification deliverable

4.4 1st Prototype Tasks

4.4.1 TaskerCLI Tasks

SE_05_TCLI_01 – Implement JUnit tests from specification

SE_05_TCLI_02 – Create application interfaces

SE_05_TCLI_03 – Implement classes from design specification and adhering to interfaces

SE_05_TCLI_04 – Implement task synchronisation logic

SE_05_TCLI_05 – Implement GUI

SE_05_TEST_01 – First prototype integration and application testing

SE_05_TCLI_06 – Performance and bug fixing

SE_05_DEL_04 – Complete first prototype deliverable

4.4.2 TaskerMAN Tasks

SE_05_TMAN_01 – Implement PHPUnit tests from specification

SE_05_TMAN_02 – Implement HTML, CSS on static content

SE_05_TMAN_03 – Implement basic editing of tasks

SE_05_TMAN_04 – Implement sorting and filtering

SE_05_TMAN_05 – Implement advanced editing of tasks (batch editing)

SE_05_TEST_01 – First prototype integration and application testing

SE_05_TMAN_06 – Performance and bug fixing

4.4.3 TaskerSRV Tasks

SE_05_TSRV_01 – Setup development database

SE_05_TSRV_02 – Create schema and indexes

SE_05_TSRV_03 – Establish connections and populate with test data

SE_05_TSRV_04 – Create automated package to setup database

SE_05_TEST_01 – First prototype integration and application testing

SE_05_TSRV_05 – Performance fixes and schema tweaks

4.5 Software delivery Tasks

SE_05_TCLI_07 – Incorporate client feedback and fixes

SE_05_TMAN_07 - Incorporate client feedback and fixes

SE_05_TSRV_06 - Incorporate client feedback and fixes

SE_05_TEST_02 – Testing for all targeted platforms and bug fixes

SE_05_MAINT _01 – Create maintenance documentation

SE_05_TCLI_08 – Package software ready for handover

SE_05_TMAN_08 – Package software ready for handover

SE_05_TSRV_07 – Package database scripts ready for handover

SE_05_PM_04 – Deliver software to client on CD

4.6 Documentation handover Tasks

SE_05_QA_01 – Review testing documentation

SE_05_QA_02 – Review design documentation

SE_05_QA_03 – Review maintenance manuals

SE_05_QA_04 – Review miscellaneous documents

SE_05_DEL_05 – Testing report

SE_05_DEL_06 – Handover all documentation

5. MONITORING

5.1 Review meetings

Review meetings will be conducted as per SE.QA.07 [8]. Before the meeting the QA manager will inform all relevant members of the team the location and time of the meeting. They will also ensure the correct version of the document is being used and distribute this copy before the meeting. During the meeting a brief version of minutes may be taken and distributed subject to a decision between the QA manager and project leader. The QA

manager will take any points that arise and create issues on GitHub and ensure they are assigned. A document must go through a review before it can have its status changed to review

5.2 Informal meetings

There will also be informal meetings held on a regular basis. These are to co-ordinate and assign tasks, check progress and raise points within the group. Two informal meetings are planned every week to allow constant communication within the group [8]. These will be held Tuesday at 6PM and Wednesday at 2PM in the Think Tank. In the event of a meeting time being changed, cancelled or moved the project manager will email the group and inform and confirm with the group the alternative arrangements. Brief minutes and actionable points will be taken by the QA manager and distributed to the group. Any issues arising will be opened and assigned on GitHub.

5.3 Formal meetings

Formal meetings will take place once per week with the project manager. In these meeting minutes must be taken by any member of the group present, and then officially typed up by the QA manager. The project manager will review minutes from the previous meeting and create and assign any actions to ensure the project continues correctly and on time. The project manager will also ensure progress has been made on previously assigned actions.

5.4 Minutes

All minutes must conform to the layout specified in SE.QA.03 [4] these minutes should be uploaded to the repository as soon as possible. Each set of minutes should be in an individual file with the name style [9] of yyyy-mm-dd_minutes where yyyy represents year, mm represents month, dd represents day in numerical form.

5.5 Time sheets

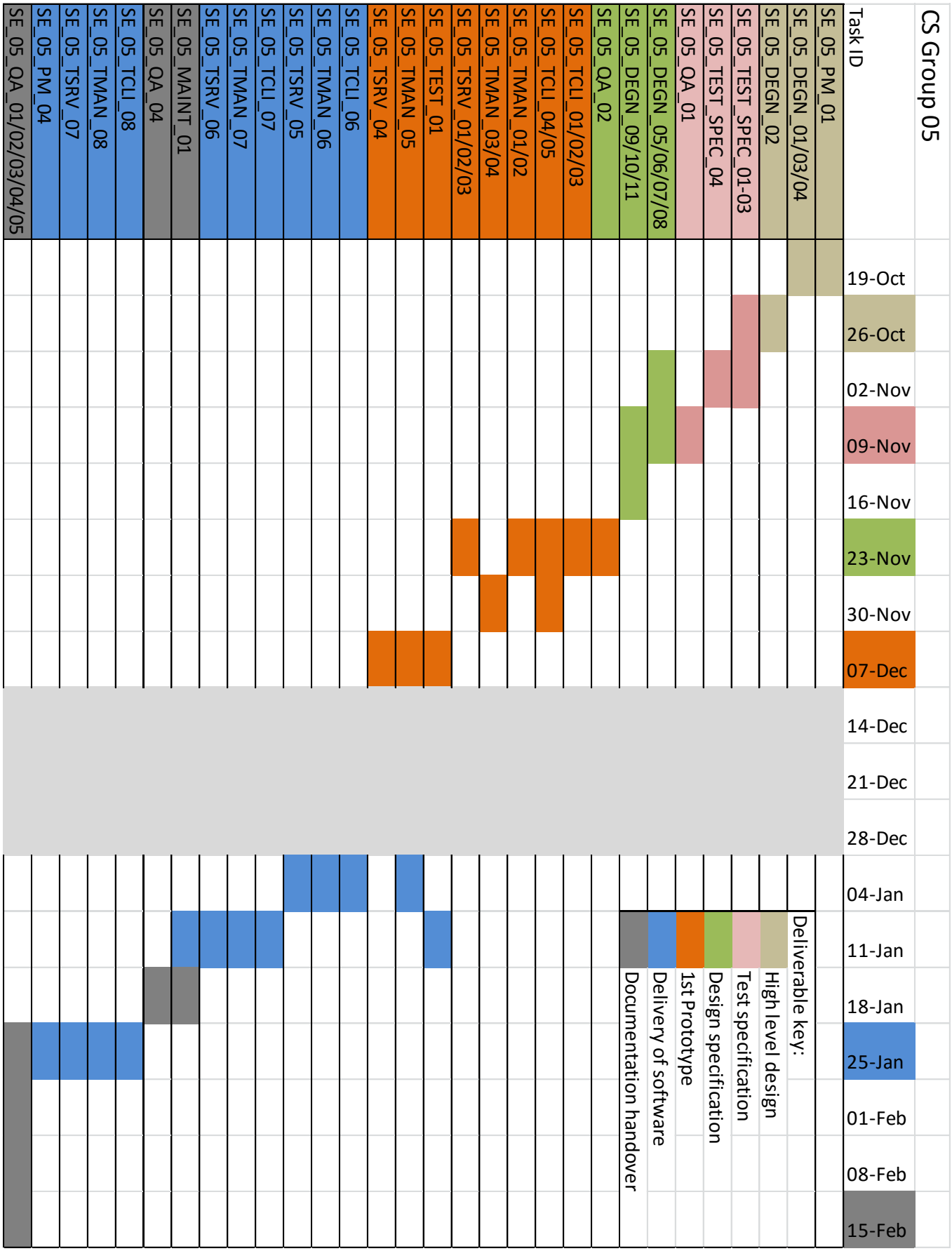
Time sheets will be created and conform to the layout provided by the project leader. Within these all project members will estimate the time required to complete a task, the time taken to complete the task, the task ID and name and any comments about the task. These will be submitted by Wednesday evening, they must show the tasks completed from the previous Wednesday to the current Wednesday.

These will be used to calculate hours available to the project, hours completed and any over or under run.

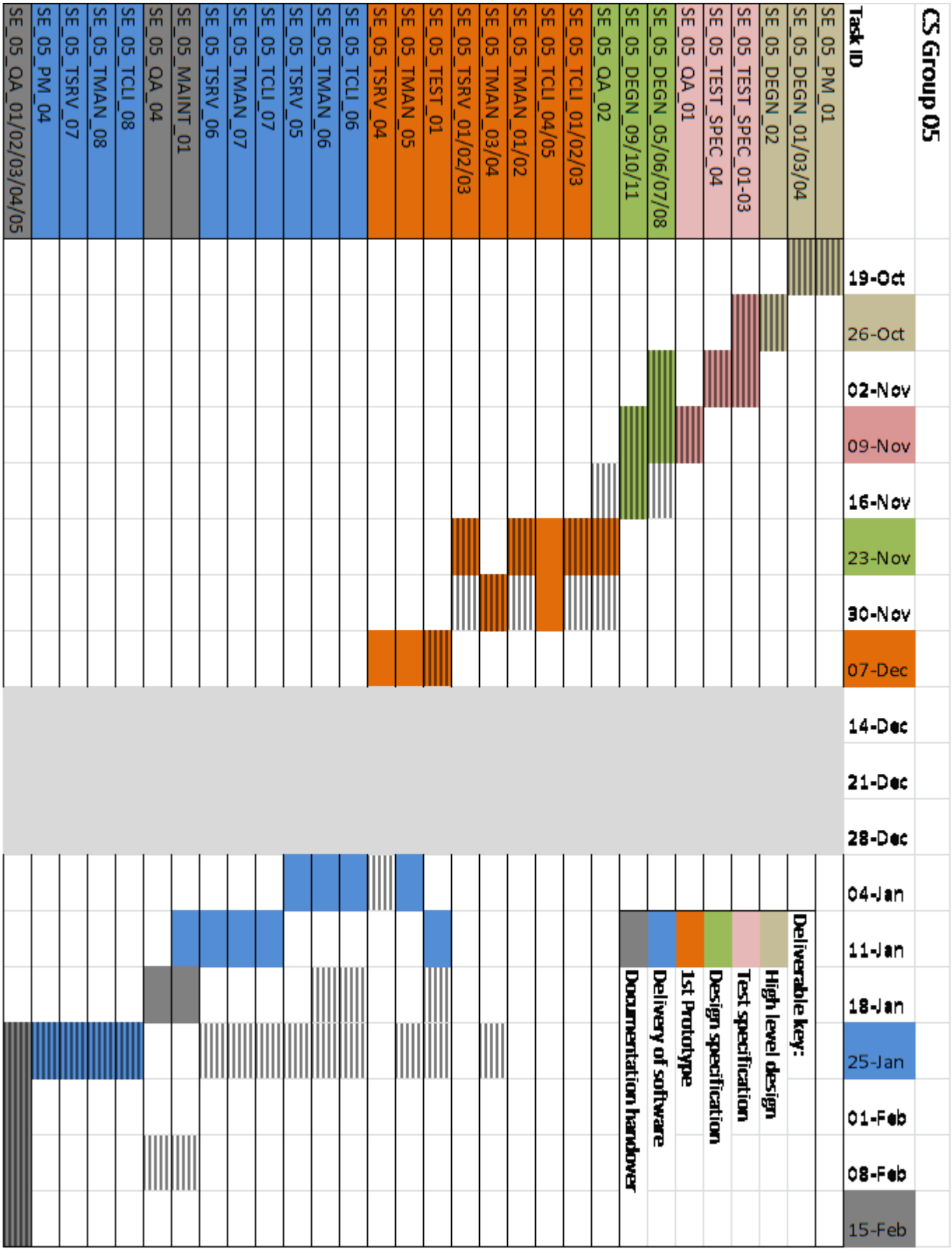
5.6 Blogs

All project members will create and maintain a blog. This will list the task they are currently working on, any comments about the task and difficulties encountered during that task. The project manager will monitor these to view the project member’s progress.

6. GANTT CHART



7. COMPLETION DATES



8. RISK ANALYSIS

Risk ID	Risk Description	Impact	Probability	Mitigation
001	Work not in on schedule	Medium	Medium	Monitoring of assigned tasks and regular feedback from team members
002	Project leader missing	Medium	Low	Deputy leader to take over when project leader missing, project leader must keep deputy leader informed
003	QA lead missing	Medium	Low	Deputy QA to take over. QA lead to keep deputy informed of documents pending review and review dates
004	Work corrupted	Medium	Low	Revert to previous commit, if still corrupted pull from server repository
005	Local work deleted	Medium	Low	Pull latest commit from GitHub and recreate local copy
006	Remote work deleted	High	Low	Revert latest commit. Upload latest version to GitHub from a team members local computer (determined in meeting) recreated repository
007	GitHub temporarily offline	Medium	Low	Watch status page at https://status.github.com/ and work locally
008	GitHub permanently offline	V. High	V. Low	Consult with project manager, migrate hosts to alternative Git host. Upload most recent local copy of work (determined in meeting)
009	Project member illness	Low-Med	Medium	Depending on severity reallocate tasks to another member, if possible get existing work off project member
010	Client requirements change	Med-High	Low	Consult regularly with client and ensure project is progressing with their satisfaction. Meeting with team and client to compromise on any requirement changes
011	GUI implementation problems	Medium	Medium	Spike work will be created early in project to create code that produces output like GUI design. In event this is not easy to implement the GUI design will be altered.
012	Client synchronisation issues	High	Medium	Spike work will be conducted early in the project to view ways of dealing with intermittent connections and sync issues

REFERENCES

[1] Aberystwyth University, "Project Timetable: The Software Development Life Cycle (2015-16)," [Online]. Available: https://blackboard.aber.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_12897_1&content_id=_557616_1. [Accessed 19 10 2015].

[2] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.05 A 1.8 - Design Specification Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

[3] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.06 1.8 - Test Procedure Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

[4] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.03 1.8 - General Documentation Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

[5] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.01 1.10 - Quality Assurance Plan,* Aberystwyth University: Software Engineering Group, 2015.

[6] C. J. Price, *SE.QA.02 - Project Management Standards 1.9,* Aberystwyth University: Software Engineering Group Project, 2015.

[7] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.1,* Aberystwyth University: Software Engineering Group Project, 2015.

[8] C. J. Price, N. W. Hardy and B. P. Tiddeman, *SE.QA.07 - Review Standards 1.6,* Aberystwyth University: Software Engineering Group Project, 2015.

[9] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.08 1.8 - Operating Procedures and Configuration Management Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	19/10/2015	First draft ready for review	DAF5
1.1	N/A	29/10/2015	Corrected spellings of JUnit and PHPUnit	JOD32
1.2	#18	30/10/2015	Corrected spelling in risk analysis and added GUI and sync logic to risk analysis	DAF5
1.3	188	13/02/2016	Added document versions to reference list	DAF5
1.4	189	13/02/2016	Added completion dates to Gantt chart	DAF5

Software Engineering Group Project
Testing Specification

Author: David Fairbrother, Maurice Corriette, Oliver Earl
Config Ref: SE_05_DEL_02
Date: 2016-02-13
Version: 2.1
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document contains a testing specification for this project. The software engineering team will use this document to ensure the working system is functionally correct, robust and matches the client's specification. Tests ensure designs are adhered to and implemented correctly and can identify more bugs within software earlier in development.

1.2 Scope

This document covers the testing specification used during development. The intended readers are the Quality Assurance Manager, Project Manager Leader and all software engineers. The testing plan and procedures are specified in SE.QA.06 [1]

1.3 Objectives

By the end of this document the reader should be informed of the nature of the tests and the methodology of testing. They should also be able to clearly identify the functional requirement [2] being tested, the purpose of that test and the expected outcome.

2. TASKERSRV / TASKERMAN TESTING

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
Please note that all TaskerMAN tests are to be carried out in both Firefox and Chrome browsers					
TS1	FR7	Logging into system	Do not enter email address and click on login button	User is told to enter an email address	User access is prohibited
TS2	FR7	Logging into system	Enter email address “antibones123” and click on login button	User is told to enter a valid email address	User access is prohibited
TS3	FR7	Logging into system	Enter email address “abcde123@example.co.uk” and click on login button	User is told that the email is not recognised	User access is prohibited
TS4	FR7	Logging into the system	Enter email address “jec17@aber.ac.uk” And click on the login button	User is told that the login was un-successful	User access is prohibited
TS5	FR7	Logging into system	Enter email address “mac81@example.co.uk” And click on login button (Test data has already been input into the database and so valid emails already exist)	User directed to main menu	User access is granted
TS6	FR7	Add new user	On the main menu click on add button Do not enter an email, first name or last name Leave ‘Is manager?’ as default Click on submit button	User is told to enter an email	Email is rejected
TS7	FR3	Add new user	On the main menu click on add button Enter email “johnsmith@example.com” Do not enter a first name, last name, leave ‘Is manager?’ as default and click on Submit button	User is told to enter a first name	First name is rejected
TS8	FR3	Add new user	Enter Email “johnsmith@example.com” First name “John Smith” Do not enter a last name, leave ‘Is manager?’ as default and click on Submit button	User is told to enter a last name	Last name is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS9	FR3	Add new user (Relevant for later test)	Enter first name “Maurice” Enter last name “Corriette” Enter email “abc123@example.co.uk” leave ‘Is manager?’ as default	Input accepted	User is added
TS10	FR3	Add new user(Relevant for later test)	Enter first name “Qi” Enter last name “XI” Enter email Qi123@example.co.uk leave ‘Is manager?’ as default	Input accepted	User is added
TS11	FR3	Add new user(Relevant for later test)	Enter first name “Josh” Enter last name “Doyle” Enter email “Doyle123@example.co.uk”	Input accepted	User is added
TS12	FR3	Add new user(Relevant for later test	Enter first name “Oliver” Enter last name “Earl” Enter email ole123@example.co.uk leave ‘Is manager?’ as default	Input accepted	User is added
TS13	FR3	Add new user	Enter first name “M” Enter last name “Tom” Enter email “abc123@example.ac.uk” leave ‘Is manager?’ as default	User is told that First name is too small	First name is rejected
TS14	FR3	Add new user	Enter name “Ma” Enter last name “T” Enter email “abc123@example.ac.uk” leave ‘Is manager?’ as default	User is told that last name is too small	Last name is rejected
TS15	FR3	Add new user	Enter first name “Maurice$£%^” Enter last name “Hardy” Enter email “Maurice@example.co.uk” leave ‘Is manager?’ as default	User is told that first name has invalid characters	First name is rejected
TS16	FR3	Add new user	Enter first name “Maurice” Enter last name “%$^” Enter email “abc123@example.ac.uk” leave ‘Is manager?’ as default	User is told that last name has invalid characters	Last name is rejected
TS17	FR3	Add new user	Enter first name “abc” Enter last name “cba” Enter email “abc123” leave ‘Is manager?’ as default	User is told that email is invalid	Email is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS18	FR3	Add new user	Enter first name “Josh” Enter Last name “Doyle” Enter email “Josh123@example.co.uk” leave ‘Is manager?’ as default	Input accepted	User is added
TS19	FR3	Add new user	Enter first name “Mark” Enter last name “Doyle” Enter email “Doyle123@example.co.uk” leave ‘Is manager?’ as default	User is told that a member with that email already exists	Email is rejected
TS20	FR3	Edit user	Edit existing user “Maurice Corriette” Change first name to blank	User is told to enter a First name	Change is rejected
TS21	FR3	Edit user	Edit existing user “Maurice Corriette” Change last name to blank	User is told to enter a Last name	Change is rejected
TS22	FR3	Edit team member	Edit existing user “Maurice Corriette” Change first name to “M”	User is told that first name is too small	Change is rejected
TS23	FR3		Edit existing user “Maurice Corriette” Change Last name to “C”	User is told that Last name is too small	Change is rejected
TS24	FR3	Edit team member	Edit existing user “Maurice Corriette” Change first name to “Maurice£$%”	User is told that first name has invalid characters	Change is rejected
TS25	FR3		Edit existing user “Maurice Corriette” Change last name to “Corriette£$%”	User is told that Last name has invalid characters	Change is rejected
TS26	FR3	Edit team member	Edit Existing user “Maurice Corriette” Change last name to “Jordan Corriette”	Input accepted	Change is successful

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS27	FR3	Edit team member	Edit existing member “Maurice Jordan Corriette” Change email to “abc123”	User is told that email is invalid	Change is rejected
TS28	FR3	Edit team member	Edit existing member “Maurice Jordan Corriette” Change email to “Doyle123@example.co.uk”	User is told that that a member with that email already exists	Change is rejected
TS39	FR3	Edit team member	Edit existing member “Maurice Jordan Corriette” Change email to “mac81@aber.ac.uk”	Input accepted	Change Is successful
TS30	FR3	Delete team member	Delete Existing member “Qi XI”	Input accepted	Deletion is successful
TS31	FR3	Delete team member	Delete existing member “John”	User is told that member does not exist	Deletion un-successful
TS32	FR4	Add new task	Do not enter task name Allocated user “Josh Doyle” Enter start date “10/11/15 Enter end date “20/11/15” Leave status as default ‘allocated’ Number of task elements “1”	User told to enter a name,	Creation of task is rejected
TS33	FR4	Add new task	Enter task name “java spike work” Do not select an allocated user Enter start date “10/11/15 Enter end date “20/11/15” Number of task elements “3” Leave status as default ‘allocated’	User is told to select an allocated user	Creation of task is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS34	FR4	Add new task	Enter task name “java spike work Allocated user “Josh Doyle” Do not select a start date Enter end date “20/11/15” Leave status as default ‘allocated’ Number of task elements “2”	User is told to enter a start date	Creation of task is rejected
TS35	FR4	Add new task	Enter task name “java spike work” Allocated user “Josh Doyle” Enter start date “10/11/15 Do not enter an end date Leave status as default ‘allocated’ Number of task elements “1”	User is told to enter an end date	Creation of task is rejected
TS36	FR4	Add new task	Enter task name “Q” Allocated user “Oliver Earl” Select start date “26/10/15” Select end date “13/11/15” Enter description “Creation of testing specification” Leave status as default ‘allocated’ Number of task elements “1”	User told that Task name is too small	Creation of task is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS37	FR4	Add new task	Enter task name "QA" Allocated user "Oliver Earl" Select start date "26/10/15" Select end date "13/11/15" Enter description "Creation of testing specification" Leave status as default 'allocated' Number of task elements "1"	User is told that task name is too small	Creation of task is rejected
TS38	FR4	Add new task	Enter task name "Testing specification" Allocated user "Maurice Jordan Corriette" Select start date "13/11/15" Select end date "23/10/15" Enter description "Creation of testing specification" Leave status as default 'allocated' Number of task elements "1"	User is told that end date must be after the start date	Creation of task is rejected
TS39	FR4	Add new task	Enter task name "QAS" Enter assigned task member "Oliver Earl" Select start date "26/10/15" Select end date "13/11/15" Leave status as default 'allocated' Number of task elements "1"	Input accepted And new modal window opened up for task elements	Task is successfully added

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS40	FR4	Add new task	Task description “test description” Task comment “test comment”	Task element accepted	Task is added successfully
TS41	FR4	Add new task (relevant for later test)	Enter task name “Testing specification” Enter assigned task member “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “1”	Input accepted And task elements modal window is opened	Task elements window opens
TS42	FR4	Add new task	Task description leave blank Task comment “test comment	User is asked to please fill out task description field	Creation of task element is rejected
TS43	FR4	Add new task	Task description “test description” Task comment leave blank	User is asked to fill out task comment field	Creation of task element is rejected
TS44	FR4	Add new task	Task description “test description” Task comment “test comment”	Input accepted	Task is successfully created
TS45	FR4	Add new task (relevant for later test)	Enter task name “PostgreSQL spike work” Allocated user “David Fairbrother” Select start date “12/10/15” Select end date “19/10/15” Number of task elements “1” Leave status as default ‘allocated’	Input accepted And Task elements window opens is opened	task elements window is opened

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS46	FR4	Add new task	Task description "T" Task comment "In Progress"	User is told that Task description is too short	Creation of task element is rejected
TS47	FR4	Add new task	Task description "TS" Task comment "D"	User is told that task Comment is too short	Creation of task element is rejected
TS48	FR4	Add new task	Task description "Research into the PostgreSQL software" Task comment "In progress"	Input accepted	Task is successfully added
TS49	FR4	Add new task (relevant for later test)	Enter task name "Expendable test" Allocated user "Maurice Jordan Corriette" Select start date "15/10/15" Select end date "25/10/15" Leave status as default "allocated" Number of task elements "-1"	User is told that number of elements cannot be 0 or less	Creation of task is rejected
TS50	FR4	Add new task (relevant for later test)	Enter task name "Expendable test" Allocated user "Maurice Jordan Corriette" Select start date "15/10/15" Select end date "25/10/15" Leave status as default "allocated" Number of task elements "0"	User is told that number of elements cannot be 0 or less	Creation of task is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS51	FR4	Add new task (relevant for later test)	Enter task name “Expendable test” Allocated user “Maurice Jordan Corriette” Select start date “15/10/15” Select end date “25/10/15” Leave status as default “allocated” Number of task elements “6”	User is told that this is too many elements and that they can add more manually later	Creation of task is rejected
TS52	FR4	Add new task (relevant for later test)	Enter task name “Expendable test” Allocated user “Maurice Jordan Corriette” Select start date “15/10/15” Select end date “25/10/15” Leave status as default “allocated” Number of task elements “1”	Input accepted And Task elements window is opened	Task elements window is opened
TS53	FR4	Add new task	Task description “£$” Task comment “TS comment	User is told that Task description characters are invalid	Creation of task elements rejected
TS54	FR4	Add new task	Task description “TS desc” Task comment “ £$%^ ”	User is told that Task comment characters are invalid	Creation of task elements rejected
TS55	FR4	Add new task	Task description three spaces Task description “TS comment	User is told that task description must consist of at least 2 letters or numbers	Creation of task elements rejected
TS56	FR4	Add new task	Task description “TS desc” Task comment three spaces	User is told that task comment must consist of at least 2 letters and numbers	Creation of task elements rejected
TS57	FR4	Add new task	Task description “TS expendable description” Task comment “Delete me maybe”	Input accepted	Creation of task is successful

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS58	FR4	Add new task	Enter task name “Testing specification” Allocated user “Maurice Jordan Corriette” Select start date “266/101/125” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “1”	User told that start date is invalid	Creation of task is rejected
TS59	FR4	Add new task	Enter task name “Testing specification” Allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “133/111/155” Leave status as default ‘allocated’ Number of task elements “1”	User is told that end date is invalid	Creation of task is rejected
TS60	FR4	Add new task	Enter task name “Testing specification” Allocated user “Maurice Jordan Corriette” Select start date “10/26/15” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “1”	User is told that start date is invalid	Creation of task is rejected
TS61	FR4	Add new task	Enter task name “Testing specification” Allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “11/13/15” Leave status as default ‘allocated’ Number of task elements “1”	User is told that end date is invalid	Creation of task Is rejected
TS62	FR4	Add new task	Enter task name “Testing specification” Allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “1”	User told that task name already exists	Creation of task is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS63	FR4	Add new task	Enter task name “£$” Select allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “13/11/15” Number of task elements “1” Leave status as default ‘allocated’	User is told that Task name contains invalid characters	Creation of task is rejected
TS64	FR4	Add new task	Enter task name “maintenance manual” Select allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “abc”	User is told that number of task elements can only consist of numbers	Creation of task is rejected
TS65	FR4	Add new task	Enter task name “maintenance manual” Select allocated user “Maurice Jordan Corriette” Select start date “26/10/15” Select end date “13/11/15” Leave status as default ‘allocated’ Number of task elements “£$%”	User is told that number of task elements can only consist of numbers	Creation of task is rejected
TS66	FR5	Edit existing task	Edit existing task “Testing specification” Change task name to blank	User told to enter a task name.	Change is rejected
TS67	FR5	Edit existing task	Edit existing task “Testing specification” Change task name to “T”	User is told that task name is too small	Change is rejected

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS68	FR5	Edit existing task	Edit existing task "Testing specification" Change task name to "£"	User told that task name has invalid characters	Change is rejected
TS69	FR5	Edit existing task	Edit existing task "QAS" Change task name to "QA"	User told that task name is invalid	Change is rejected
TS70	FR5	Edit existing task	Edit existing task "Testing specification" Change assigned task member to "Cartman"	User told that assigned task member does not exist	Change is rejected
TS71	FR5	Edit existing task	Edit existing task "Testing specification" Change start date to "266/100/155"	User is told to enter a valid start date	Change is rejected
TS72	FR5	Edit existing task	Edit existing task "Testing specification" Change end date "133/111/155"	User is told to enter a valid start date	Change is rejected
TS73	FR5	Edit existing task	Edit existing task "Testing specification" Add a task element task description "A" Task comment "test comment"	User is told that task description must be two characters or more	Change rejected
TS74	FR5	Edit existing task	Edit existing task "Testing specification" Add a task element Task description "AS" Task comment "T"	User is told that task comment must be two characters or more	Change rejected
TS75	FR5	Edit existing task	Edit existing task "Testing specification" Add a task element task description "AB" Task comment "AC"	Input accepted	Change is successful

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS76	FR5	Edit existing task	Edit existing task "Testing specification" Change task name to "Design specification"	Input accepted	Change is successful
TS77	FR5	Edit existing task	Edit existing task "Design specification" Change assigned task member to "Josh Doyle"	Input accepted	Change is successful
TS78	FR5	Edit existing task	Edit existing task "Design specification" Change start date to "9/11/15"	Input accepted	Change is successful
TS79	FR5	Edit existing task	Edit existing task "Design specification" change end date to "7/11/15"	User told that end date cannot be before the start date	Change is rejected
TS80	FR5	Edit existing task	Edit existing task "Design specification" Change end date to "30/11/15"	Input accepted	Change is successful
TS81	FR5	Edit existing task	Edit existing task "Design specification" Change start date to "01/12/15"	User is told that the start date cannot be after the end date	Change is unsuccessful
TS82	FR5	Edit existing task	Edit existing task "Design specification" Change description to "Creation of design specification"	Input accepted	Change successful

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS83	FR5	Edit existing task	Edit existing task “Design specification” Change task name to “PostgreSQL spike work”	User told that task name already exists	Change is rejected
TS84	FR3	Delete existing member	Delete member “Oliver Earl”	User is told that Member cannot be deleted as he still has task(s) assigned	Delete is unsuccessful
TS85	FR6 / FR3	Abandon task	Select existing task “QAS” and click on abandon button	In the status columns for the task “Abandoned” should be present”	Task is abandoned
TS86	FR3	Delete existing member	Delete member “Oliver Earl”	Input accepted	Deletion is successful
TS87	FR4	Delete task elements	Select existing task “Test specification” and delete task elements	No output	Deletion successful
TS88	FR4	Checking for deleted task elements	Select existing task “test specification” and view task elements	No task elements should be visible on task elements modal window	Task “Test specification” has no task elements
TS89	FR4	Delete task elements that do not exist	Select existing task “test specification” and click on delete elements	User is told that task does not have any tasks	Deletion is unsuccessful

Test Ref	Req being tested	Test content	Input	Output	Pass criteria
TS90	FR6	Abandoning task	Select existing task "PostgreSQL spike work" Then click on the abandon button	In the Status column for the task, "Abandoned" should be present	Task is abandoned
TS91	FR6	Edit an abandoned task	Select abandoned task "PostgreSQL spike work" Click on Edit button	User is told task cannot be edited as it has been abandoned	Ability to edit is denied
TS92	FR6	Delete task	Select allocated task "Expendable task" And click on the delete button	User told that task cannot be deleted as it is still allocated and has not yet been abandoned	Deletion of task is unsuccessful
TS93	FR6	Abandon task	Select allocated task "Expendable task" and click on the abandon button	In the Status column for the task, "Abandoned" should be present	Task is abandoned
TS94	FR6	Abandon task	Select abandoned task "Expendable task" and click on delete button	Dialogue box appears asking user to confirm deletion	Task is deleted
TS95	FR7	View list of tasks	Gain access to main menu	Task "PostgreSQL Spike work" and "Design specification" are present in the list.	Tasks are visible and correct in the list

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS96	FR7	Check list order	Gain access to menu	All Tasks should be ordered by their end date	List is in correct order filtered by end date
TS97	FR7	Check list order	Gain access to menu Filter order via clicking on status column	All tasks should be ordered by status. Allocated being the highest, followed by completed followed by abandoned.	List in correct order sorted by status Allocated ranked higher than abandoned

3. TASKERCLI TESTING

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS98	FR8	Logging into system	Open desktop application TaskerCLI Do not enter email address and click on login button	User is told to enter an email address	User access is prohibited
TS99	FR8	Logging into system	Enter email address “antibones123” and click on login button	User is told to enter a valid email address	User access is prohibited
TS100	FR8	Logging into system	Enter email address jjj123@aber.ac.uk and click on login button	User is told that the email is not recognised	User access is prohibited
TS101	FR8	Logging into system	Enter email address “Mac81@aber.ac.uk” And click on login button	User directed to main menu	User access is granted
TS102	FR8a	Tasks specific to login in user	Look through tasks specifically the assigned users	There should be no tasks that has a user assigned other than Maurice Jordan Corriette	Only tasks assigned to Maurice Jordan Corriette are visible
TS103	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL”db.dcs.ac.uk” Enter Port number “3306” Enter Database name “csgp_5_15_16” Enter Database username “csgpadm_5” Enter database password “906BnQjD” Then press on the connect button	Coloured Circle should remain Red and error message should appear saying ”Error: Invalid Database URL entered”	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	Req being tested	Test content	Input	Output	Pass criteria
104	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL”” Enter Port number “3306” Enter Database name “csgp_5_15_16” Enter Database username “csgpadm_5” Enter database password “906BnQjD” Then press on the connect button	Coloured Circle should remain Red and error message should appear saying ”Error: Please enter a Database URL”	TaskerCLI Synchronisation with TaskerSRV Unsuccessful
TS105	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL”db.dcs.ac.uk” Enter Port number “330006” Enter Database name “csgp_5_15_16” Enter Database username “csgpadm_5” Enter database password “906BnQjD” Then press on the connect button	Coloured Circle should remain Red and error message should appear saying ”Error: Invalid port number entered”	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS106	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.ac.uk" Enter Port number "" Enter Database name "csgp_5_15_16" Enter Database username "csgpadm_5" Enter database password "906BnQjD" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: Please enter a port number"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful
TS107	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.ac.uk" Enter Port number "3306" Enter Database name "csgp_5_15_16" Enter Database username "csgpadm" Enter database password "906BnQjD" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: invalid Database username"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS108	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.ac.uk" Enter Port number "3306" Enter Database name "csgp_5_15_16" Enter Database username "" Enter database password "906BnQjD" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: Please enter a Database username"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful
TS109	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.aber.ac.uk" Enter Port number "3306" Enter Database name "" Enter Database username "csgpadm_5" Enter database password "906BnQjD" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: Please enter a Database name"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	Req being tested	Test content	Input	Output	Pass criteria
110	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.aber.ac.uk" Enter Port number "3306" Enter Database name "csgp_5_15" Enter Database username "csgpadm_5" Enter database password "906BnQjD" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: invalid Database name"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful
111	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.aber.ac.uk" Enter Port number "3306" Enter Database name "csgp_5_15_16" Enter Database username "csgpadm_5" Enter database password "" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: Please enter Database password"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	Req being tested	Test content	Input	Output	Pass criteria
112	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL"db.dcs.aber.ac.uk" Enter Port number "3306" Enter Database name "csgp_5_15_16" Enter Database username "csgpadm_5" Enter database password "906adm" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: invalid Database password"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful
113	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL "abc" Enter Port number "abc" Enter Database name "abc" Enter Database username "abc" Enter database password "abc" Then press on the connect button	Coloured Circle should remain Red and error message should appear saying "Error: invalid Database details"	TaskerCLI Synchronisation with TaskerSRV Unsuccessful

Test ref	**Req being tested**	**Test content**	**Input**	**Output**	**Pass criteria**
TS114	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button.	Coloured Circle should be green and “Connected” should be clearly visible in bold.	TaskerCLI Synchronisation with TaskerSRV Succesful
TS115	FR11	Checking for start-up synchronisation	Gain access to main menu and then click on the Connection setting button. Enter Database URL”db.dcs.aber.ac.uk” Enter Port number “3306” Enter Database name “csgp_5_15_16” Enter Database username “csgpadm_5” Enter database password “906BnQjD” Then press on the connect button	Coloured circle should Turn green And “Connected” should be clearly visible in bold	TaskerCLI successfully syncs with TaskerSRV
TS116	FR11	Checking for periodic synchronisation	Following previous test, close connection setting page and leave TaskerCLI running for 5 minutes Then reopen the connection settings page.	On the connection setting page last synced should be “5 minutes ago”	TaskercLI successfully syncs with database after 5 minutes
TS117	FR8a	Checking for local storage of tasks	Go to “TaskerCLI program directory” and then access subdirectory “data” Open “Design specification” txt file	User should be able to view data of Design specification	Local storage of Design specification data confirmed

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS118	FR10	Select task to view	Gain access to main menu and select the task 'Design specification'	In quick view section of main menu, 'Design specification' details including task name, status , assigned task member, start/end date and task elements should be shown	Tasks are selectable and can be viewed in quick view
TS119	FR10	Select multiple tasks to view	Gain access to main menu and select the tasks 'Design specification' and 'PostgreSQL spike work'	In quick view section of main menu, 'Design specification' Details should be shown. Click on the > icon and the second selected task 'PostgreSQL spike work' should be now viewable	Selected tasks can both be viewed in quick view using the < > buttons
TS120	FR10	Edit task status	Edit task 'Design specification' Change task status to 'Completed' Then click save	Input accepted	Edit is successful
TS121	FR10	Edit task status	Edit task 'Design specification' Change task status to 'Abandoned' Then click save	User is told that he/she cannot set task to abandoned	Edit is rejected

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS122	FR10	Edit task step comments	Edit task 'Design specification' Change task elements to Blank Then click save	User told to give a description for the task	Edit is rejected
TS123	FR10	Edit task step comments	Edit task 'Design specification' Change task elements to 'Task has now been completed' Then click save	Input accepted	Edit is successful
TS124	FR10	Edit task step comments	Edit 'Design Specification' Change task elements to '£$%^&' Then click save	User is told that Task elements cannot contain only special characters	Edit is unsuccessful
TS125	FR11	Checking synchronisation timing (Test to be carried out immediately after TS…)	Access main menu and click on Connection setting button	Connection setting window should display "last connected less than 1 minutes ago	
TS126	FR9	Checking synchronisation of local storage of data and that of TaskerSRV	Following changes made to Design specification Access TaskerMAN And view Design specification	Status should now be "Completed" And task elements should be "Task has now been completed"	Confirmation of synchronisation between local storage of data and TaskerSRV through TaskerMAN

Test ref	Req being tested	Test content	Input	Output	Pass criteria
TS127	FR11	Checking synchronisation timing (Test to be carried out immediately after TS…)	Access main menu and click on Connection setting button	Connection setting window should display “last connected less than 1 minutes ago	TaskerCLI synced immediately after task update
TS128	FR9	Checking synchronisation of local storage of data and that of TaskerSRV	Following changes made to Design specification Access TaskerMAN And view Design specification	Status should now be “abandoned”	TaskerCLI Successfully synced with TaskerSRV following edit of Design specification
129	FR8a	Checking for local storage of tasks	Disconnect the computer from the internet temporarily. Then edit pre-existing task “Test report” changing the status from allocated to completed.	The change should be made on TaskerCLI and be evident	Change has been made on TaskerCLI
130	FR8a	Checking for local storage of tasks	Reconnect the computer to the internet and connect TaskerCLI to the database via appropriate credentials , as follows: Enter Database URL”db.dcs.ac.uk” Enter Port number “3306” Enter Database name “csgp_5_15_16” Enter Database username “csgpadm_5” Enter database password “906BnQjD”	On connection settings page, and main menu “Connected” should be visible	TaskerCLI reconnects and syncs with database
131	FR8a	Checking for local storage of tasks	Log into TaskerMAN with username “manager@example.com” And view task “test specification”	The status of task “test report” should be set as completed	TaskerCLI automatically updates database with locally stored changes.

REFERENCES

[1] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.06 1.8 - Test Procedure Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

[2] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
0.1	N/A	2015-11-5	Original Version	MAC81
0.2	N/A	2015-11-8	Document up for review	MAC81
0.3	N/A	2015-11-11	Corrected Template	DAF5
1.0	N/A	2015-11-14	Document Release	MAC81
2.0	196	2016-02-13	Complete update of document	DAF5
2.1	188	2016-02-13	Added document versions to references	DAF5

Software Engineering Group Project
Design Documentation

Authors:	Ben Dudley, David Fairbrother, Jonathan Englund, Josh Doyle, Liam Fitzgerald, Maurice Corriette, Oliver Earl, Tim Anderson
Config Ref:	SE_05_DEL_03
Date:	13/02/2016
Version:	2.0
Status:	Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to describe and specify a full design for all software within this project. This will be used by software engineers to implement several components. It also lists the integration of these components to facilitate a solution matching the client's requirements [1].

1.2 Scope

This document shows a complete and full design for TaskerCLI, TaskerMAN and TaskerSRV. It lists the requirements to run each component and how they integrate and communicate to each other. It goes on to list how these components will be implemented specifying in detail algorithms where required.

1.3 Objectives

This document contains a complete view of the software solution designed. Initially the design considers high level aspects of design such as typical use cases and GUI mock ups it. It then lists classes used within these various components which are further broken down in complex classes into activity diagrams, sequence diagrams and spike work conducted. The reader should be able to visualise the overall look of the software whilst understanding the implementation at lower levels of design.

2. DEPLOYMENT DESCRIPTION

2.1 Applications in the system

2.1.1 TaskerCLI

TaskerCLI is the desktop based application in the system. The software will be written in Java and will be tested with Java 1.7.0_85 running on a Linux 64-bit Operating System. [Appendix A] - using versions of the Java Runtime Environment lower than this may cause unexpected behaviour and therefore is not recommended.

JDBC will be used to facilitate data communication. The version this software will be developed with is 4.2, utilising driver version 5.1.37.

The JUnit testing framework that is used during development will be version 4.12. This requires Java Development Kit 1.5 or above. [1]

2.1.2 TaskerMAN

TaskerMAN is the web-based software component of the system. The website will be built with HTML5, CSS (Cascading Style Sheets), JavaScript and PHP. The PHP tested during development is PHP Version 5.6.13 [Appendix B] running on an Apache server [Appendix C], running on Gentoo Linux 3.18.7 64-bit [Appendix D].

This information is also available by running *phpinfo()* on the targeted web server. [2]

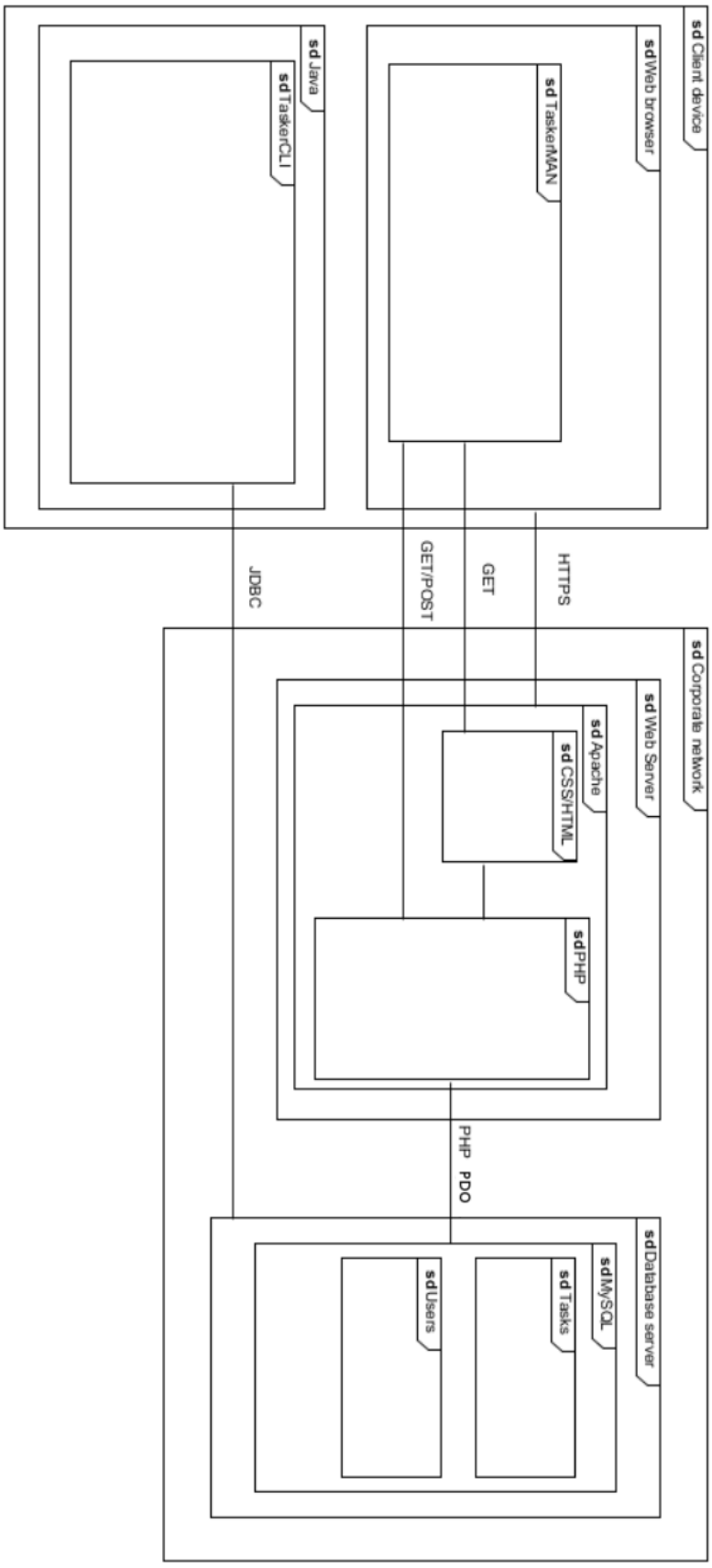
In order to enable the use of the PHPUnit testing framework, a minimal installation of PHP 5.6 is required, but the latest install is highly recommended. [3]

2.1.3 TaskerSRV

TaskerSRV is the database component.

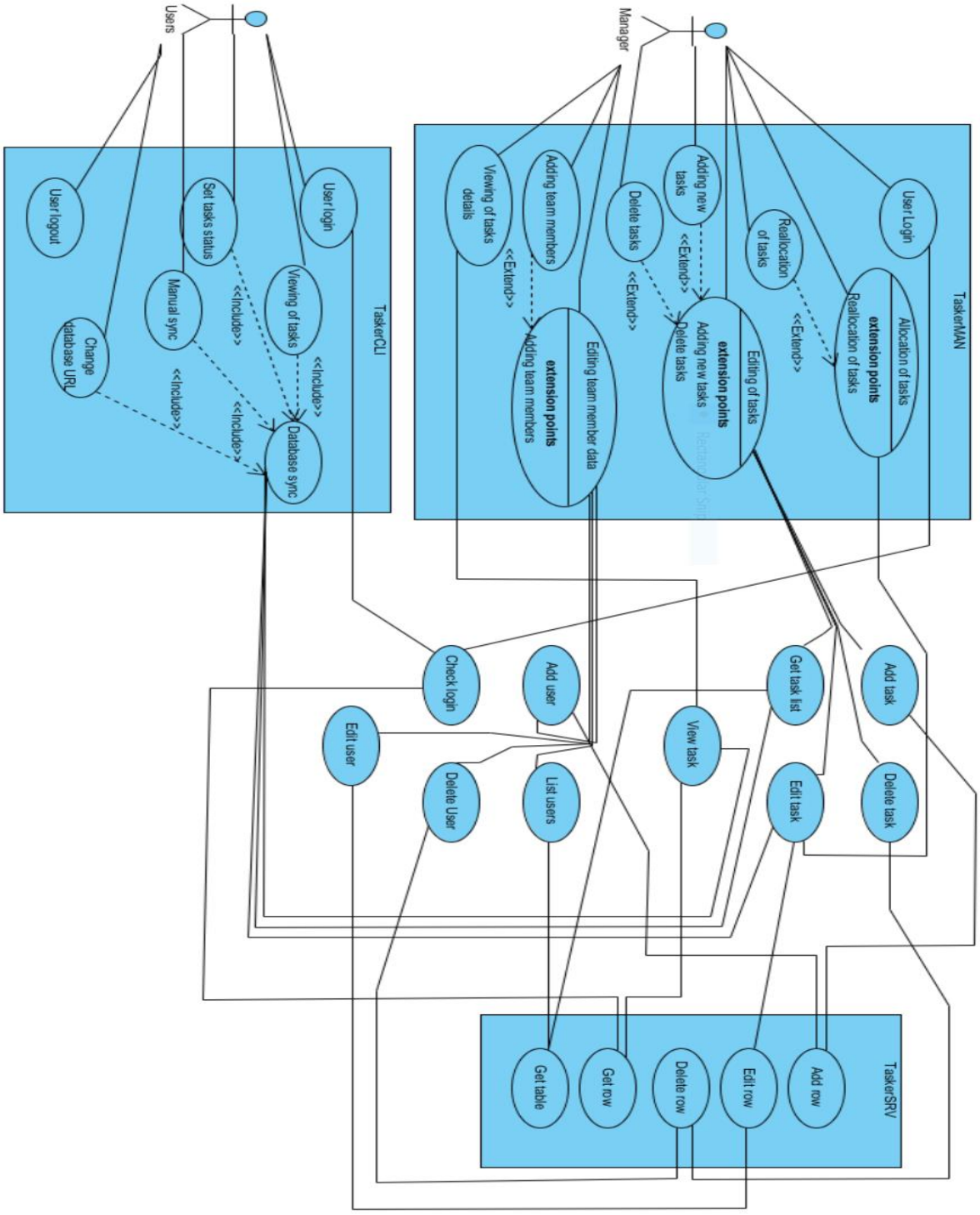
A MySQL relational database will be used. The version will be tested against is MySQL 5.6.26 on a Linux 64-bit Operating System [Appendix E]. The main system requirement for a current MySQL installation is 2.5GB of free hard disk space [4], and any disk space pertinent to the size of the database.

2.2 Application interactions



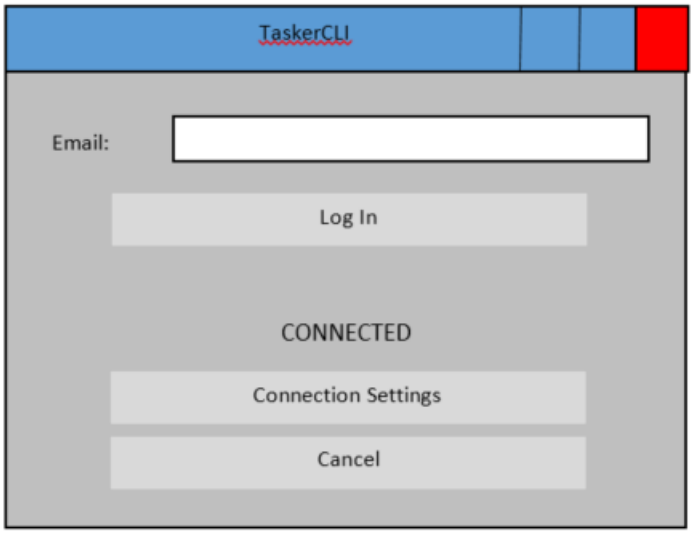
3. INTERACTION DESIGN

3.1 Use-Case Diagrams



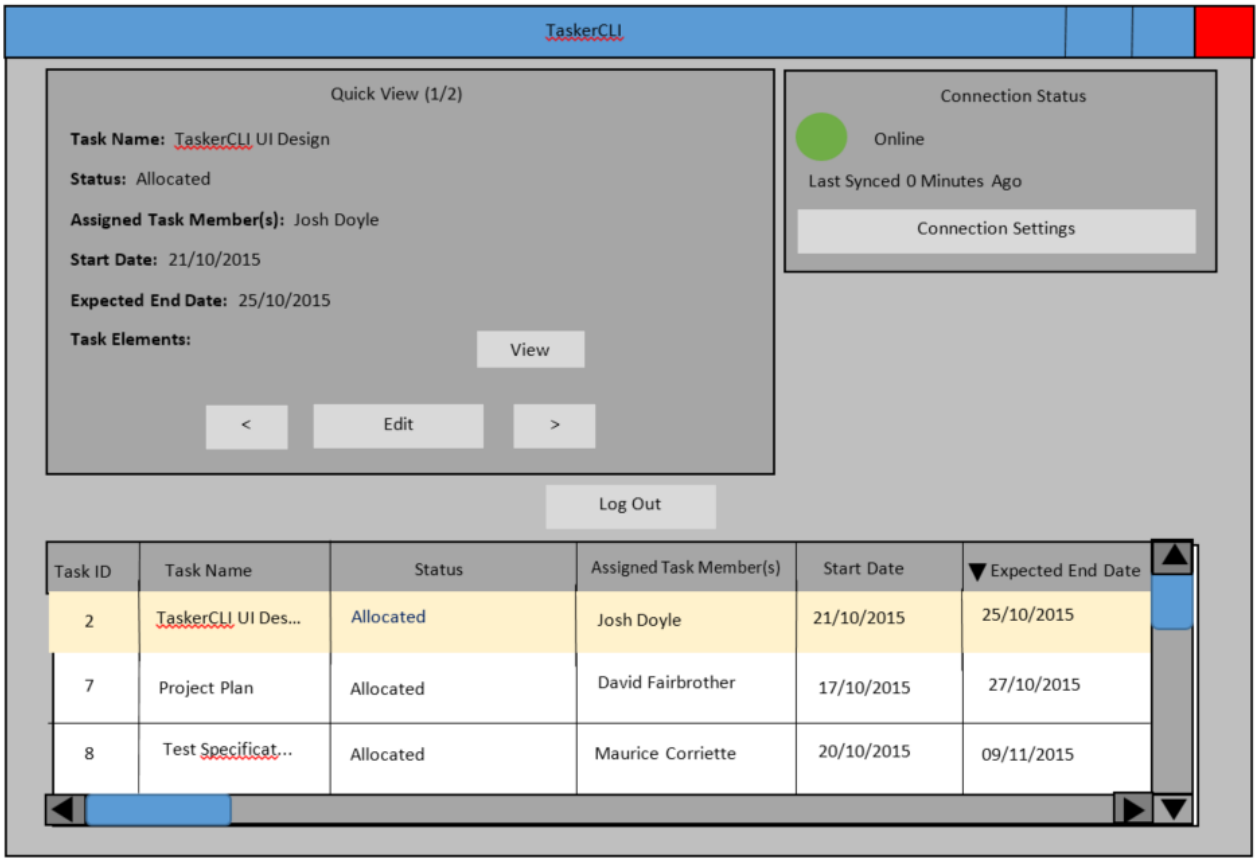
3.2 User Interface Design – Tasker CLI

3.2.1 Log In Window



- Textbox for user to enter their email address for validation, as per requirements specification. [6]
- Log In button opens ‘Main Window’ when clicked, provided a valid email address has been entered.
- Clicking the ‘Connection Settings’ button will open the ‘Connection Settings’ window – this allows the user to configure their connection to the *TaskerSRV* database.
- Closing ‘Log in Window’ will bring up the ‘Exit Confirmation’ window.

3.2.2 Main Window

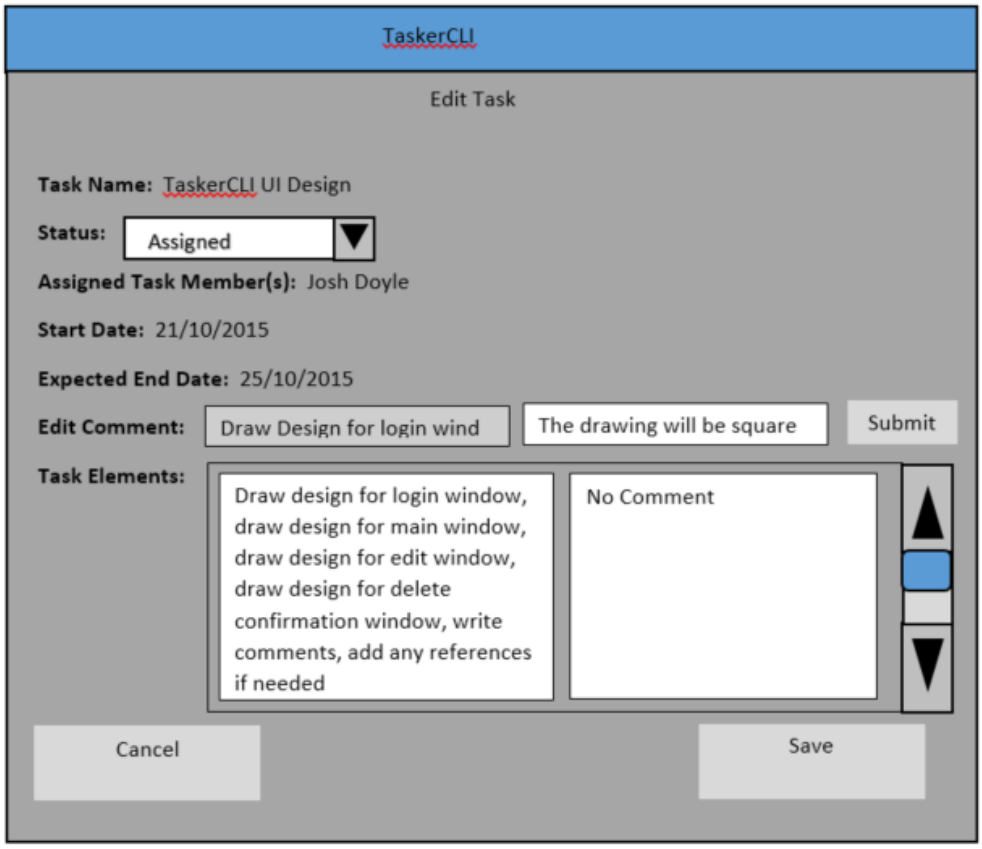


- The table at the bottom of the 'Main' window shows all of the tasks currently saved in *TaskerSRV*, the last time the program was synchronised with the database.
    - Clicking on the headings at the top of the columns in the table, will order the table based upon the values in that column. The design shown is in descending order based upon the *Expected End Date* column.
    - Checking the checkboxes next to each task, enables the user to select multiple tasks.
    - The scrollbars are used to navigate the table.
    - Selected tasks are shown in more detail in the 'Quick View' panel.
- The 'Quick View' panel at the top left of the 'Main' window presents the data from the tasks selected from the table.
    - Clicking the arrow keys at the bottom of the 'Quick View' panel navigates between all tasks selected from the table at the bottom of the 'Main' window.
    - Clicking the 'Edit' button opens the 'Edit' window, to change completion status and task elements of the current task in the 'Quick View' panel.
- The Connection Status panel at the top right of the 'Main' window status changes colour depending on the connection status.
    - Green indicates that *TaskerCLI* is currently connected to *TaskerSRV* and that everything is synchronised.
    - Red indicates that the connection between *TaskerCLI* and *TaskerSRV* has been lost and that synchronisation is no longer guaranteed.

- In the demonstrated design, *TaskerCLI* is connected to *TaskerSRV* and has been synchronised less than a minute ago. The number of minutes increments every minute and returns to 0 after successful synchronisation.

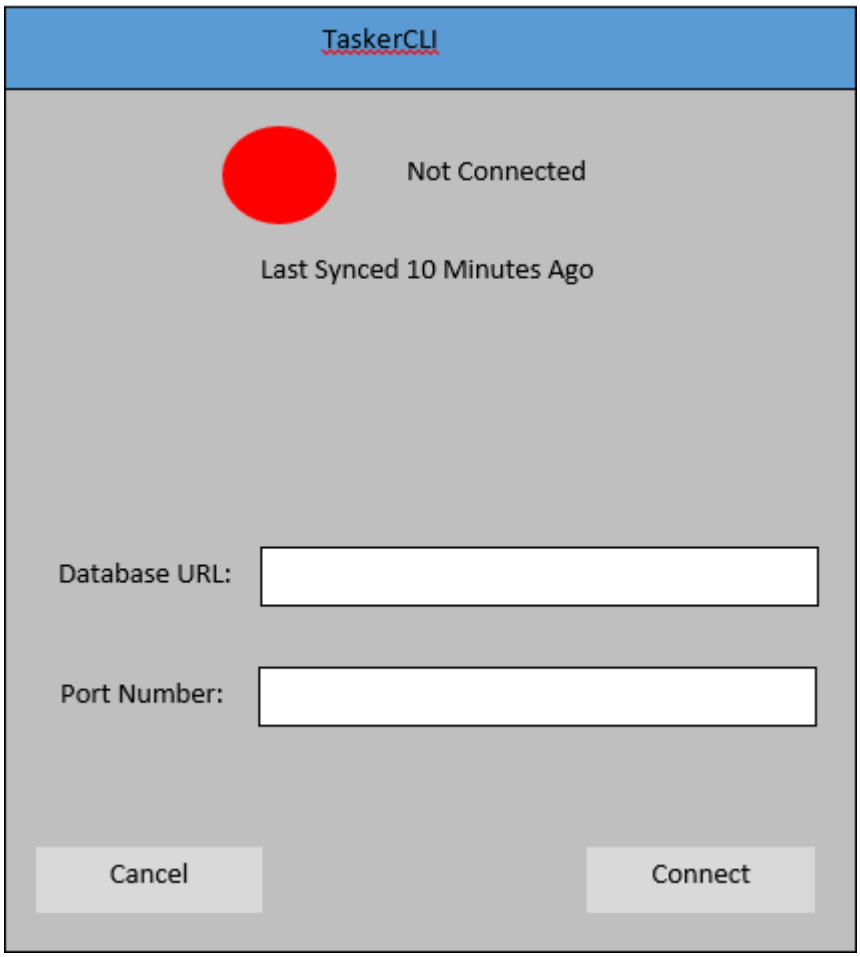
- Closing the window brings up the 'Exit Confirmation' window.

3.2.3 Edit Window



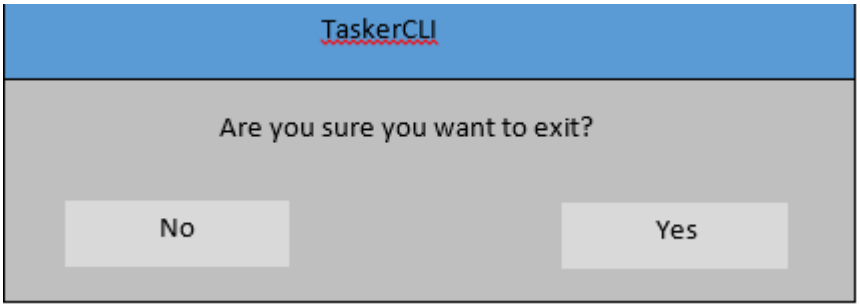
- The 'Edit Task' window is populated with the data of the task that was in the 'Quick View' panel on the 'Main' window when it was opened.
- The attributes of completion status and task elements are editable from this window.
- The completion status can be selected from a dropdown list. The default value is 'Assigned.'
- The task elements can be changed by typing into the Task Elements textbox.
    - A scrollbar will only appear if the text entry exceeds the size of the textbox.
- When the Save button is clicked, the 'Edit Task' window is closed and the task attributes are updated with their new values.
- Choosing Cancel simply closes the window with no changes.

3.2.4 Connection Settings Window



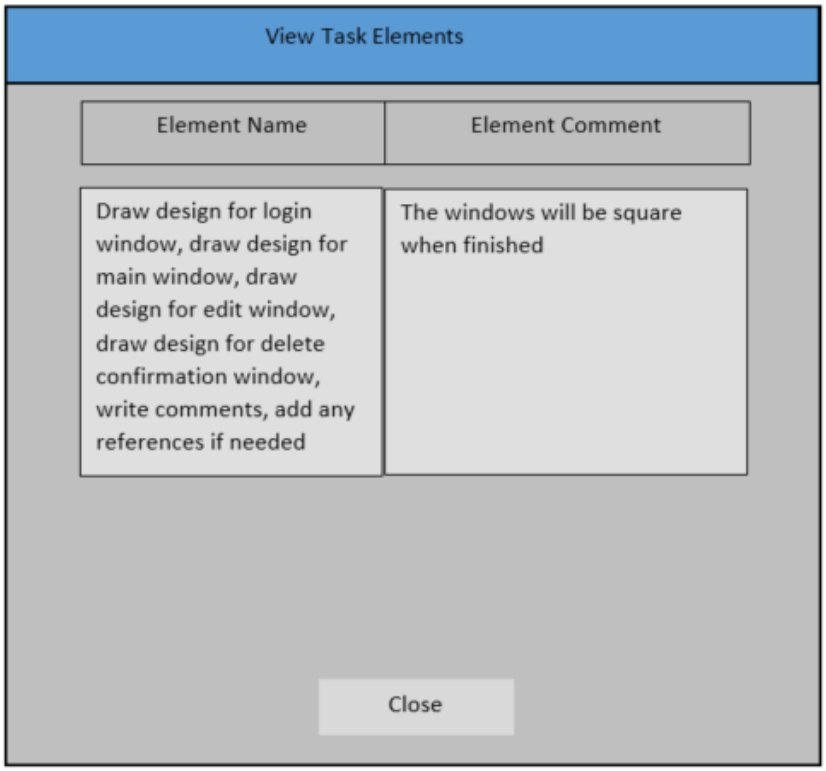
- The connection status text at the top of the window shows the current connection state of *TaskerCLI*.
  - The coloured circle is red when there is no connection established to *TaskerSRV*.
    - Consequently the coloured circle appears green when a connection is successfully established.
  - The time since last sync shows how much time has passed since the last synchronisation.
  - In this design, *TaskerCLI* is not connected to *TaskerSRV* and it has been 10 minutes since the last successful synchronisation.
- The Database URL and Port Number are entered into the respective fields to provide information for connecting to the *TaskerSRV* database.
- Choosing 'Cancel' simply closes the window without saving any information.
- Choosing 'Connect' will instruct *TaskerCLI* to attempt to connect using the information provided.
- Default window controls and clicking outside of the window are disabled to prevent the user from opening multiple instances of this window and attempting to cause simultaneous connections to be established.

3.2.5 Exit Confirmation Window



- If 'No' is selected, the window is closed and the user regains control of the window they were previously using.
- If 'Yes' is selected, *TaskerCLI* closes.
- Default window controls are disabled to make it clear to the user that their attention is required and that a decision must be made.
- Clicking away from the window to bring another window into focus is also disabled, to stop the user spawning multiple instances of the 'Exit Confirmation' window.

3.2.6 View Tasks Window



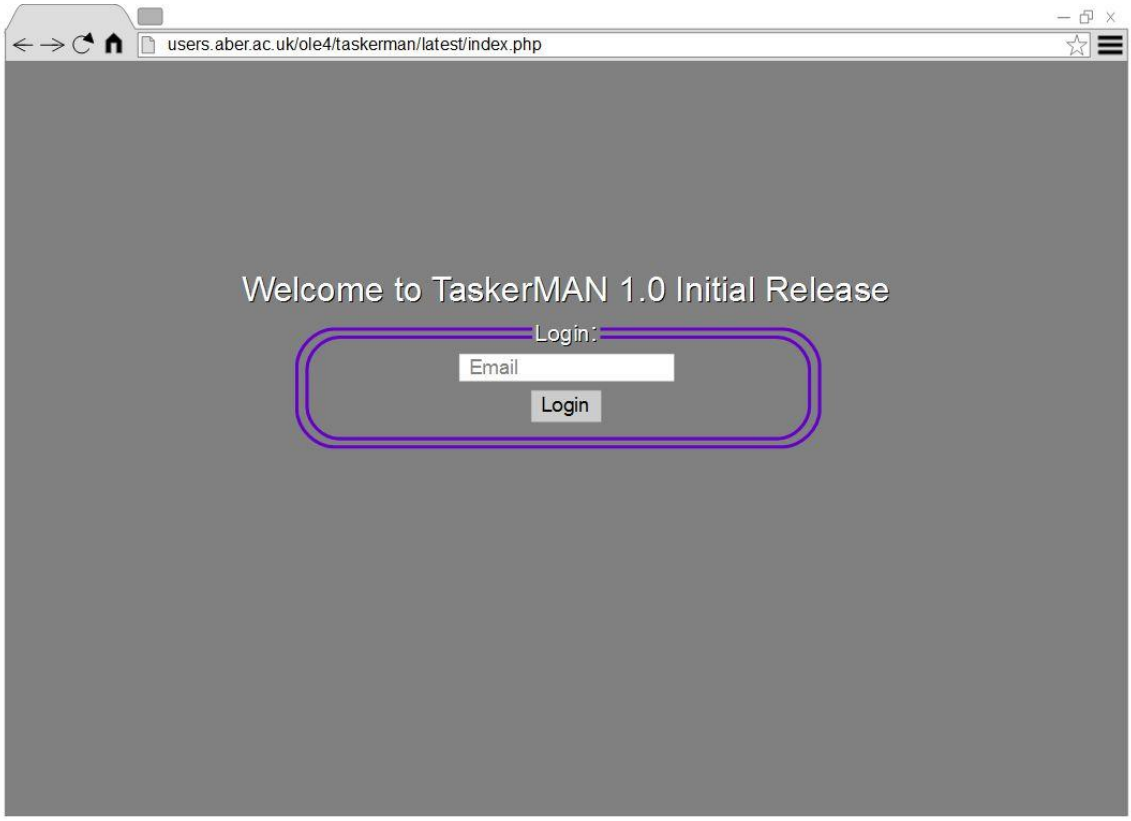
- Grid layout of element and comment pars displayed
- If no elements are associated with task display ‘No Element’ and ‘No Comment’
- Multiple element comment pairs will display in table format.

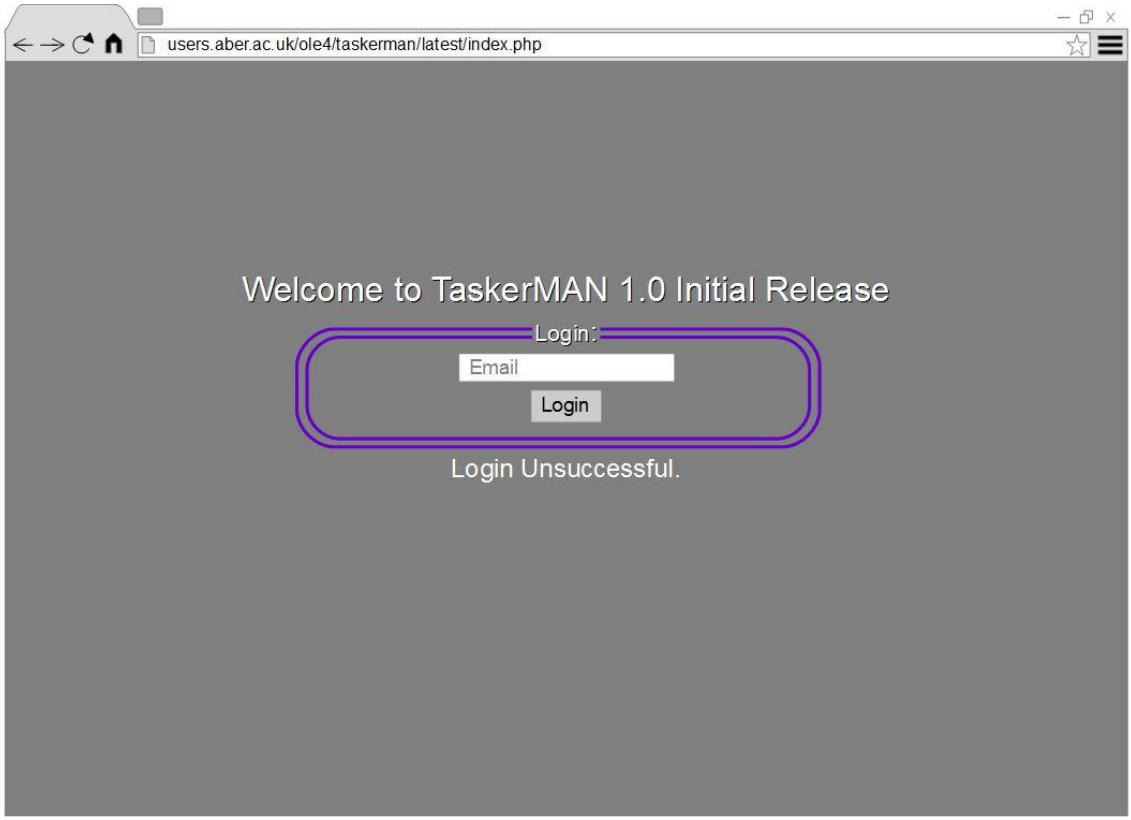
3.3 User Interface Design - TaskerMAN

3.3.1 General Notes

Mozilla Firefox is used as an example web browser in the following images. [6]

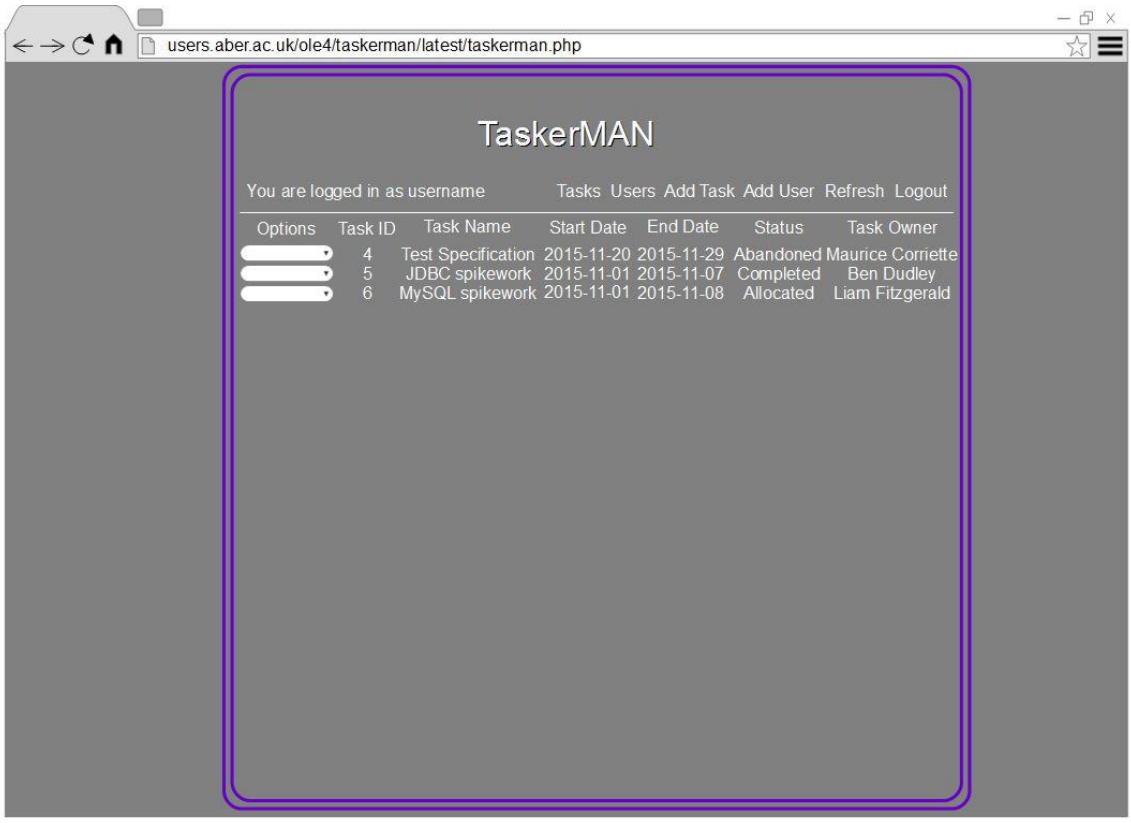
3.3.2 Login Page

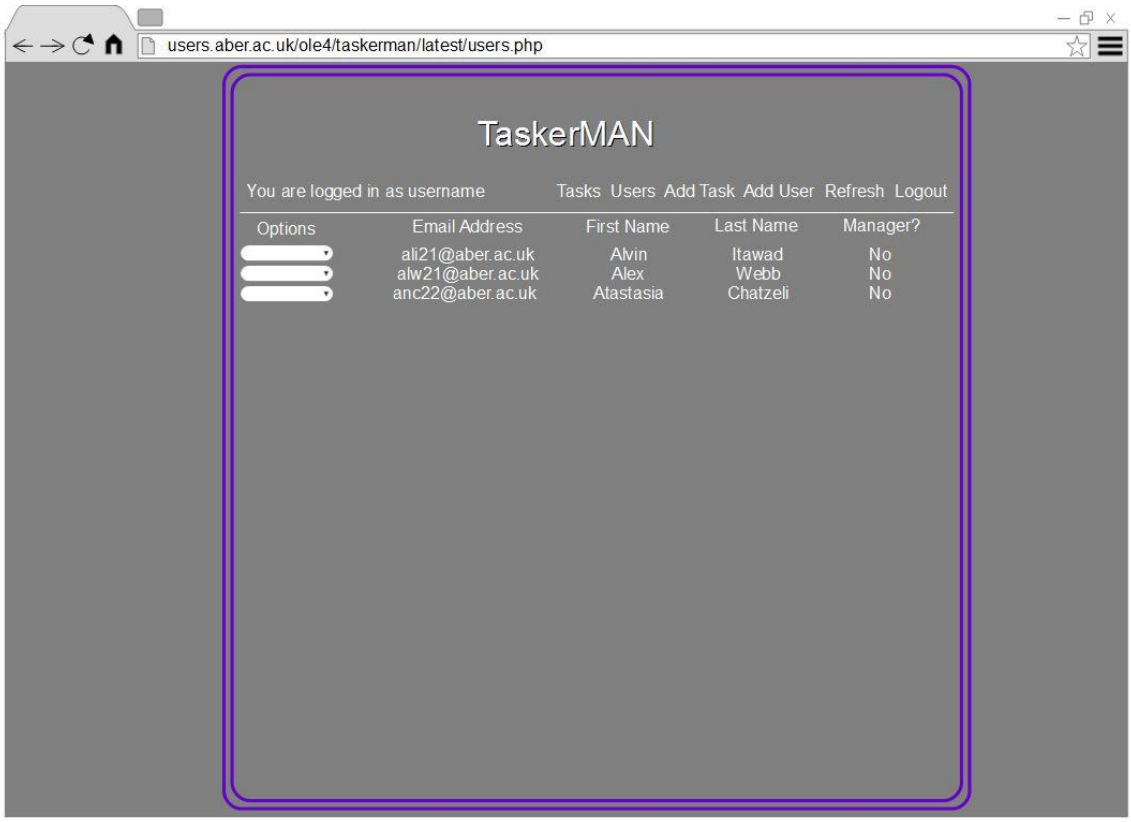




- If a valid email address is entered, the user will be directed to the main page – otherwise access is prohibited, as is required. [6]
- It is also required that the email address belongs to a manager. If this is not the case the user will be denied access.

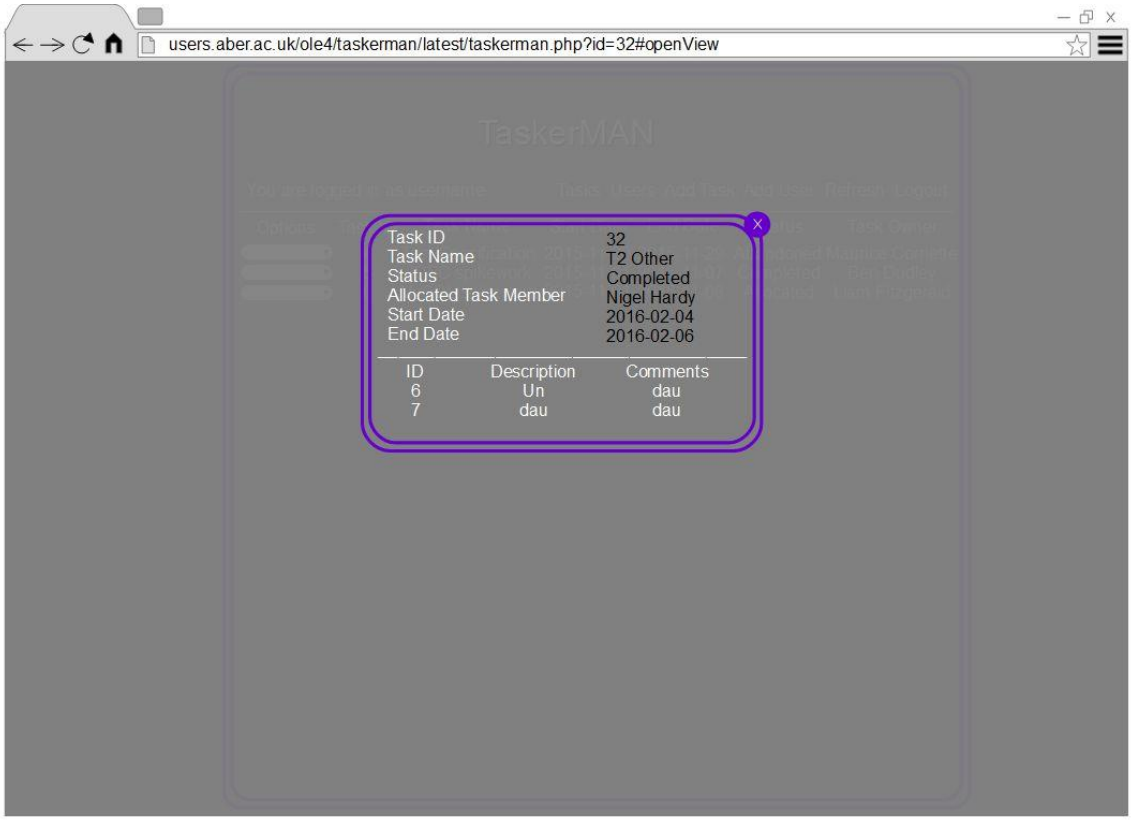
3.3.3 TaskerMAN Task / User View





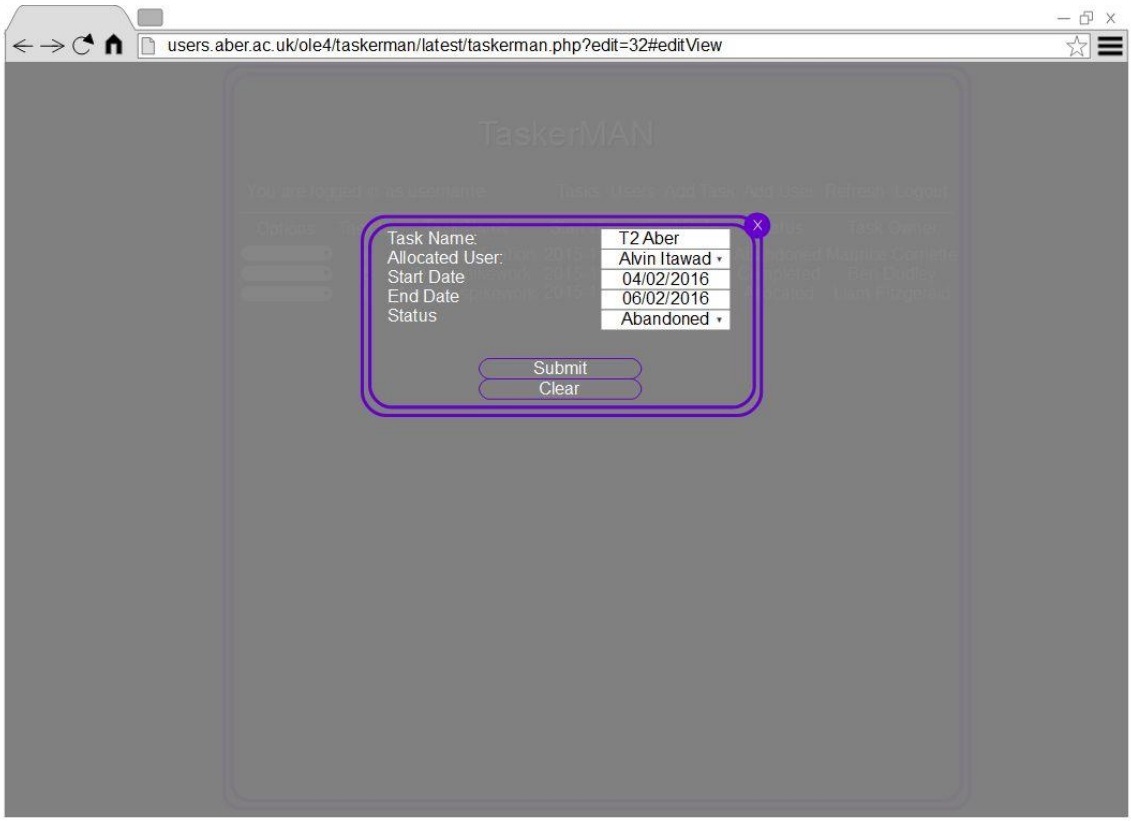
- Displays the user's name at the top of the screen, indicating who is currently logged in.
- Tasks and Users change the current TaskerMAN view to the other options.
- Add Task displays the modal window to add a new task to the database.
- Add User displays the modal window to add a new user to the database.
- Drop down next to individual users and tasks to allow users to view a user, task or its elements and edit them.
- Refresh reloads the TaskerMAN interface and retrieves any new data from the database.
- The Logout link simply logs the user out of the system, redirecting them to the Login page.

3.3.4 View Elements / View Task



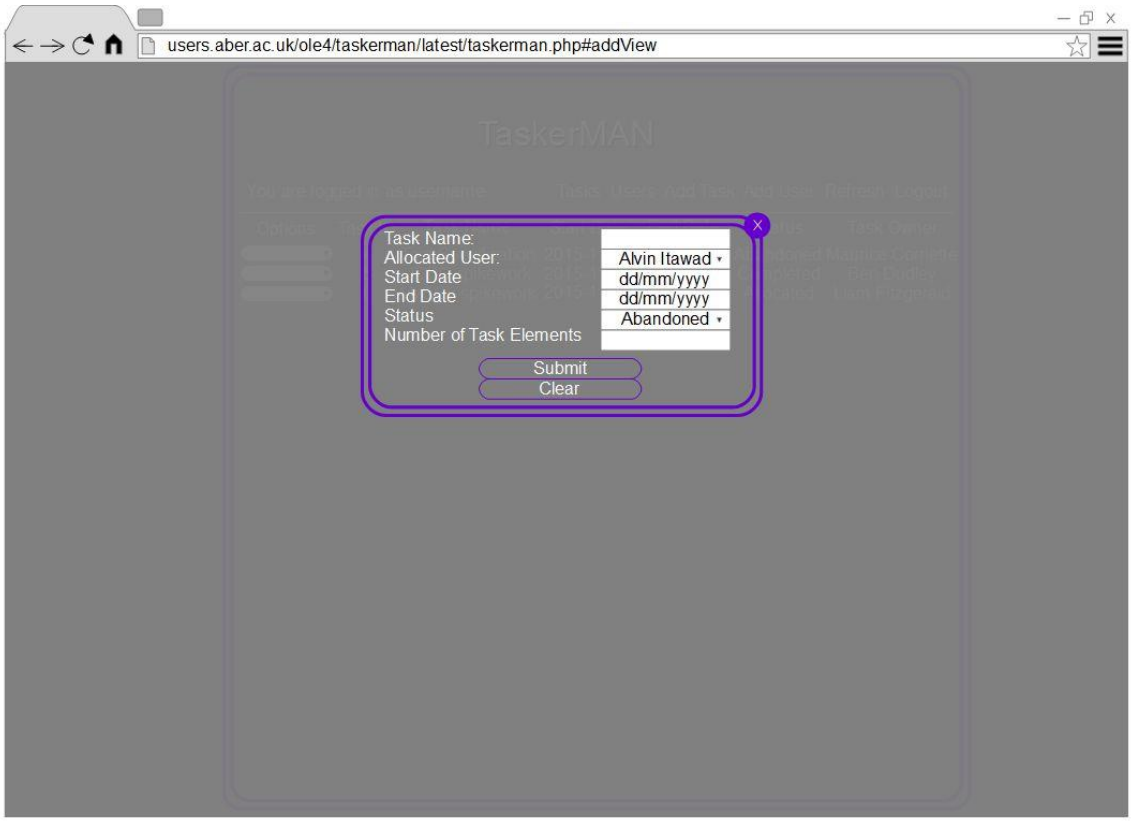
- Displays selected tasks elements.
- Read-only – data is not editable here.

3.3.5 Edit Task Overlay



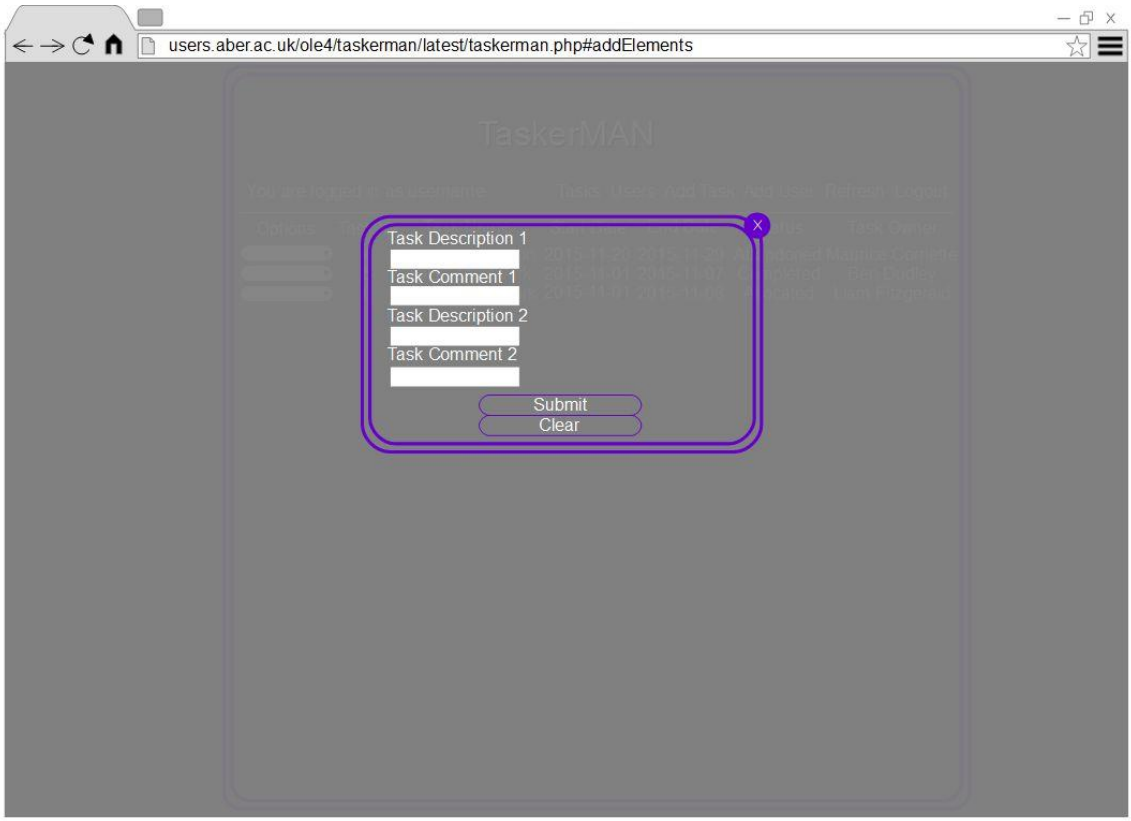
- All fields here are editable, allowing complete editing of the task.
    - It does not allow the editing of elements as these are contained in a different window.
    - Data is validated and sanitised to prevent invalid input.
- Submit will submit the data. Clear blanks the form.
- Closing the modal window will cancel any changes.
- The database will be updated after each edit, refreshing TaskerMAN in the process.

3.3.6 Add Task Overlay



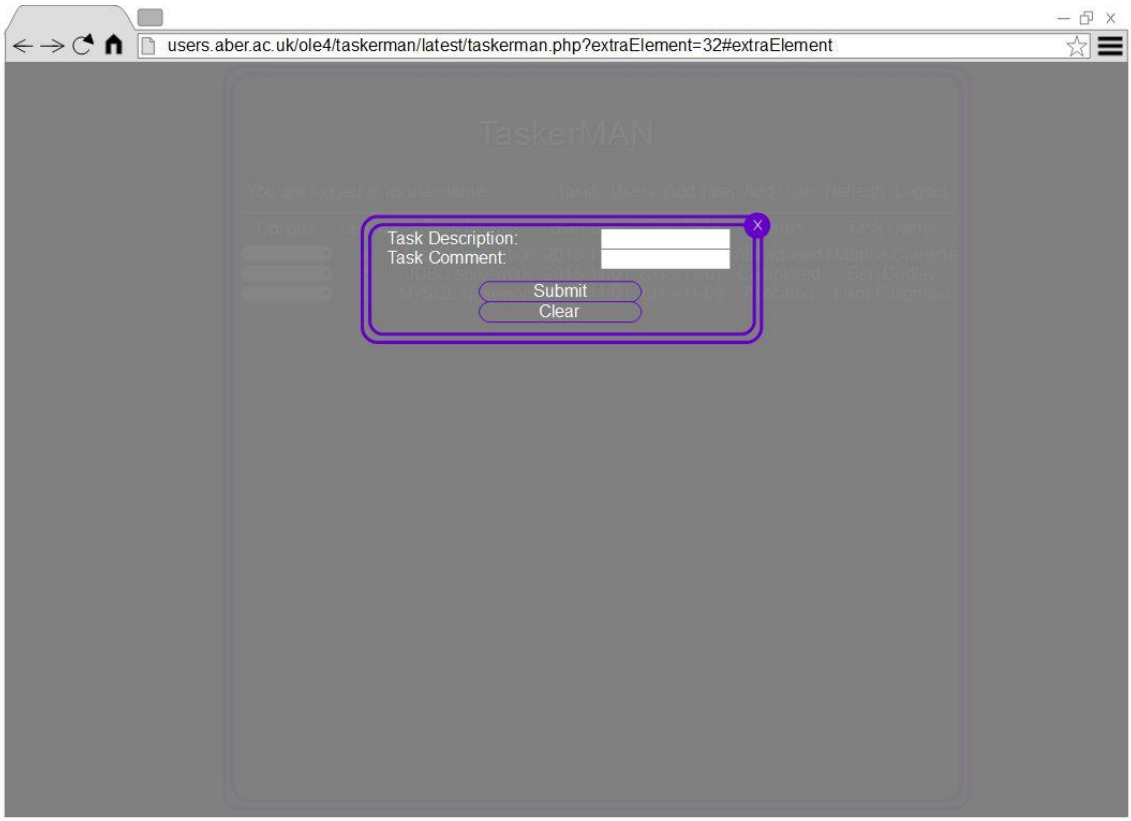
- In visual likeness to the 'Edit Task' screen, except blank task details and allows for a completely new entry to be added.
- Option for number of task elements allows the user to add more elements on another modal window that appears afterwards.
- Validation and sanitisation will be used to ensure only correct/meaningful data can be entered.

3.3.7 Add Tasks – Elements Window



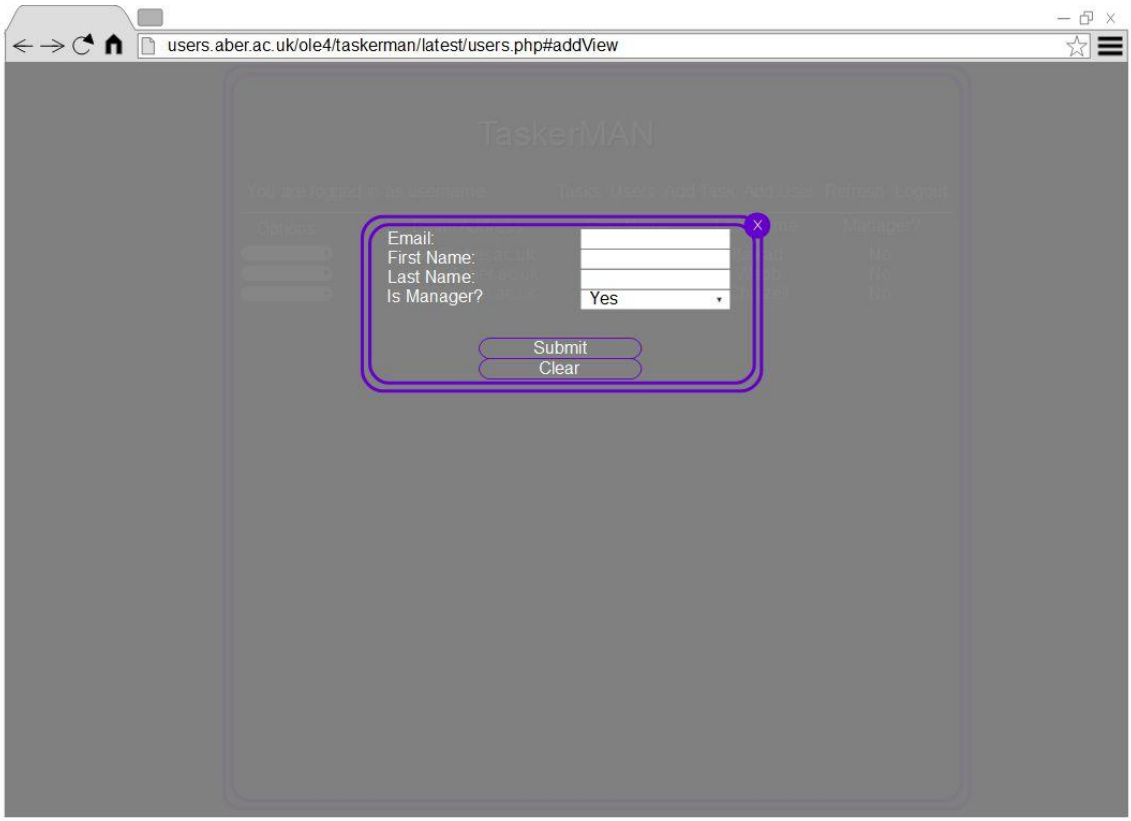
- Number of text input boxes for task element's descriptions and comments is determined by the number of task elements inputted in the previous window.
- This is capped at five elements at a time.
- Inputs are validated and sanitised to ensure valid input.
- The user can add additional elements later via the Add Element option.

3.3.8 Add Element



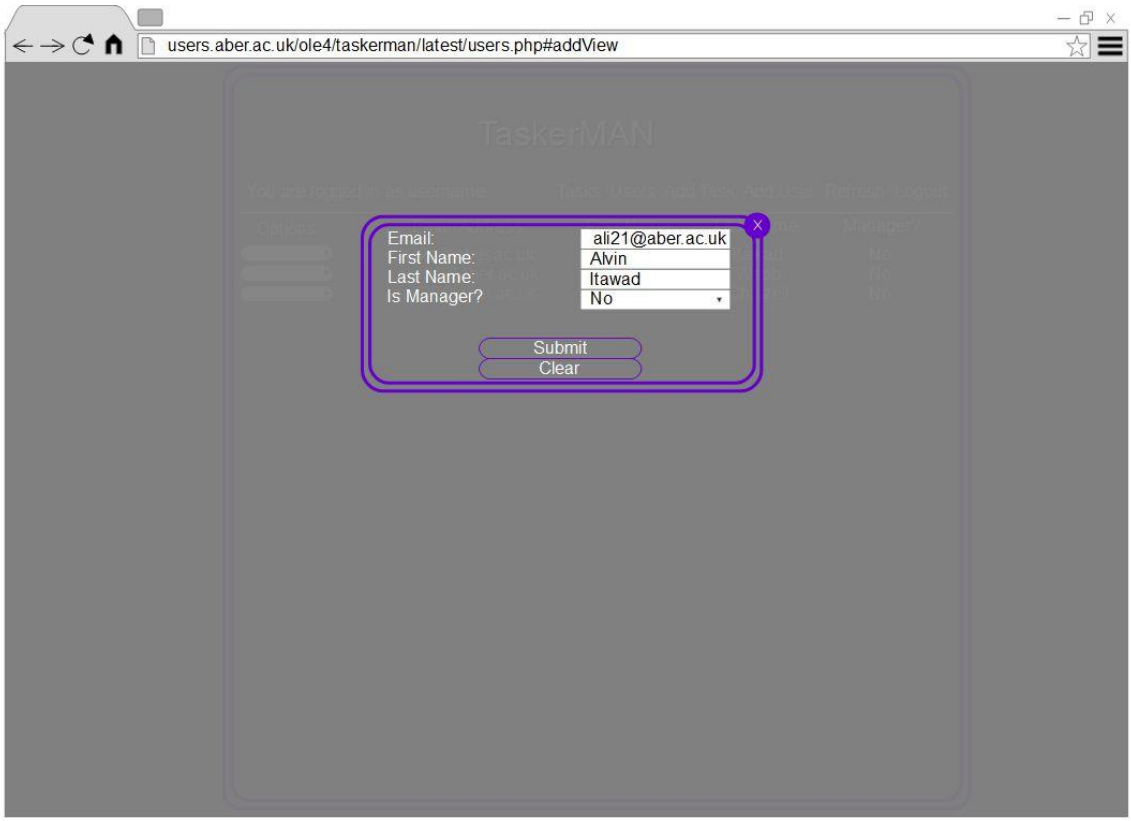
- Allows adding a single extra task – both the description and task comment
- Sanitised and validated to ensure valid input

3.3.9 Add User



- Allows adding a new user to the system including setting their managerial status
- Validated and sanitised

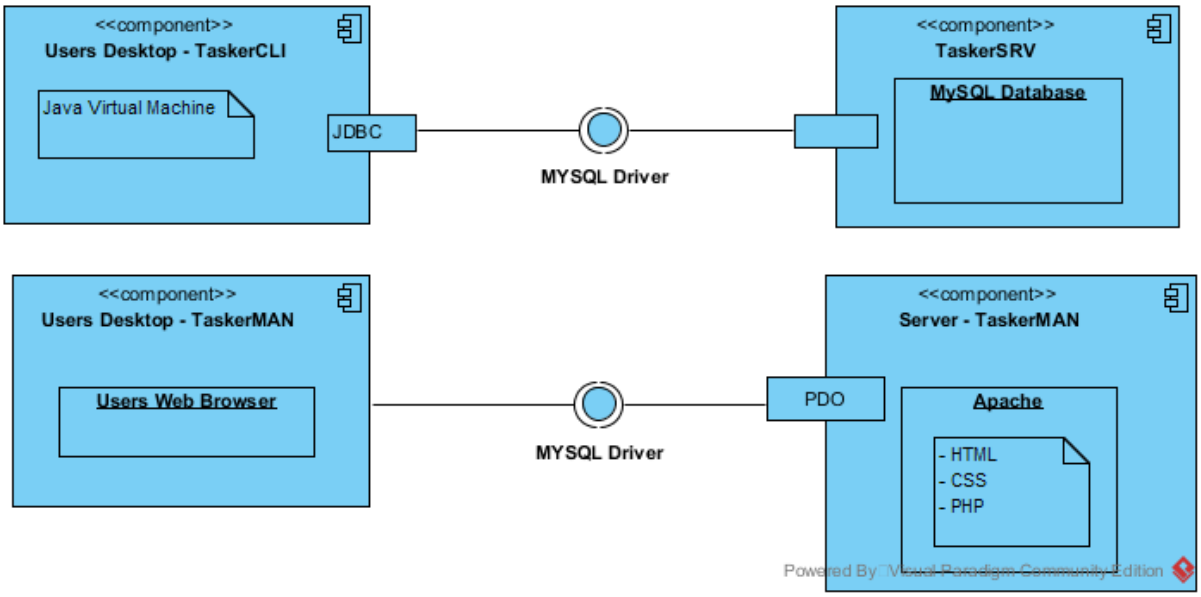
3.3.10 Edit User



- Allows user to edit an existing user in the system
- This includes modifying their managerial status
- Validation and sanitisation are used to ensure data is valid

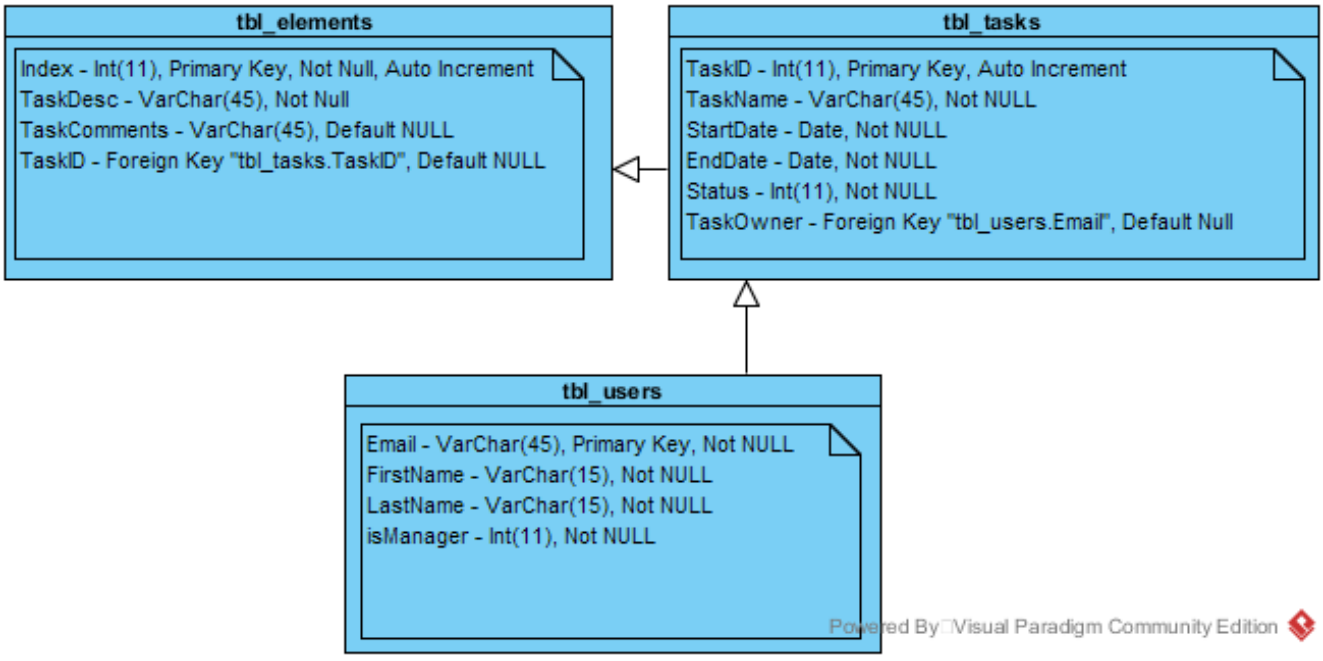
4. COMPONENT DESCRIPTION

By using standard libraries to connect various components to TaskerSRV we can utilise standard protocols provided by the MYSQL driver avoiding the need to specify interfaces for inter component communication. Clients will initiate the connection to TaskerSRV and handle the connection through JDBC and PDO for TaskerCLI and TaskerMAN respectively.



4.1 TaskerSRV Database Design

Using this design the database in TaskerSRV must use a standard naming scheme and have fixed properties. These are listed in the diagram below.



5. SIGNIFICANT CLASSES

5.1 TaskerCLI

TaskerCLI classes can be broken down into functional groups. Classes which handle data including editing, database synchronisation and ordering are grouped as “Logic Classes”. The remaining Classes are used to power the GUI such as getting user inputs and displaying or closing windows, these are grouped as “GUI Classes”

5.1.1 Logic Class Diagram

See Appendix F – Logic Class Diagram.

The classes and descriptions are as follows:

Database: - Holds a JDBC connection and performs execution of SQL statements in order to both send and receive data to TaskerSRV.

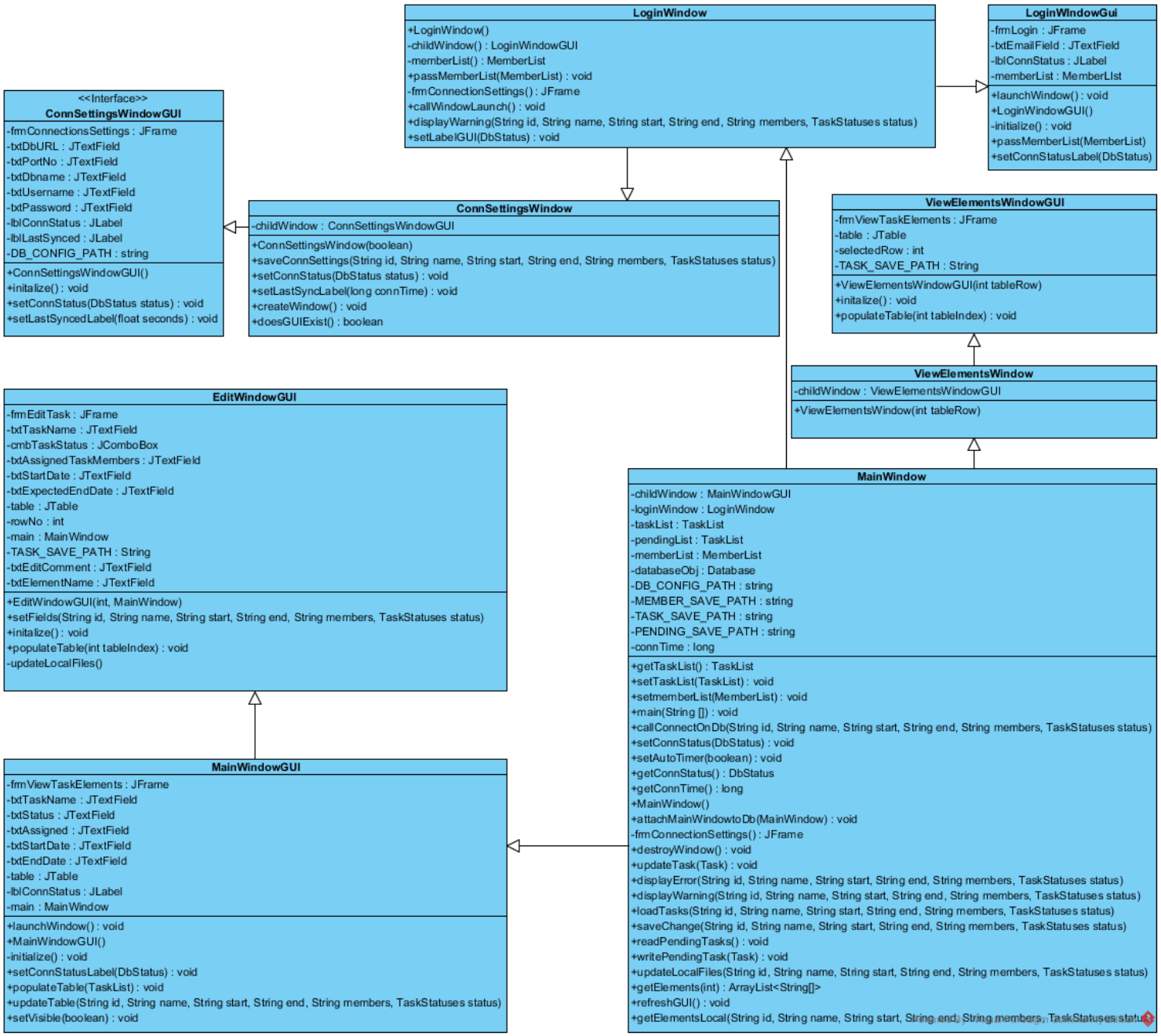
Task: - Represents an individual task and holds the name, elements, dates and assigned users of a task.

Member: - Represents an individual user of the system. Contains the users email address and name.

MemberList: - A class which holds all members found in TaskerSRV as Member objects.

TaskList: - A class which holds all tasks found in TaskerSRV as Task objects.

5.1.2 GUI Class Diagram



5.1.3 Logic Classes Interface List

Task:
String getID();
String getName();
String getStart();
String getEnd();
String getMembers();
String getStatus();
Void setStatus(TaskStatuses newStatus);
Void addElement(String elementName, String elementComment, String index);
Void clearAllElements();
Element getElement(int index);
ArrayList<Element> getAllElements();

Member:
String getName();
String getEmail();
void setName();
void setEmail();

MemberList:
Members getMember(int index);
Void addMember(Members member);
Void loadMembers(String filename);
Boolean memberExists(String email);
Int getLength();

TaskList:
ArrayList<Task> getTaskList();
void addTask(Task task);
void setAssignedTasks();
Task getTask(int index);
int getListSize();
void changeTask(int taskPos, Task newTask);

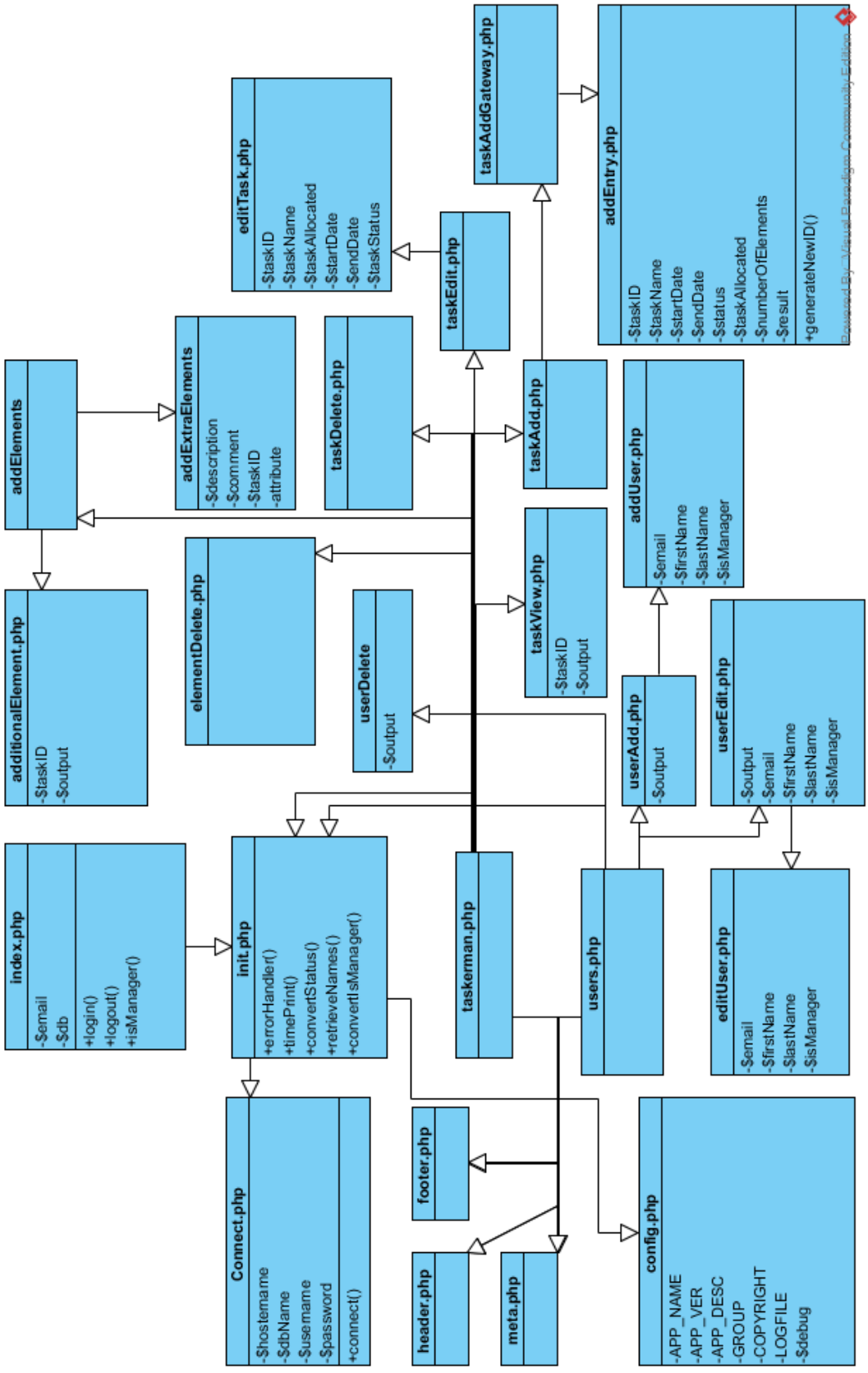
5.1.4 GUI Classes Interface List

Window:
void populateWindowArray();
void initialize (windowIndex, windowName);
void exit(windowIndex, windowName);
void setFocus(windowIndex, windowName);

5.2 TaskerMAN

5.2.1 Class Diagram

TaskerMAN is written in procedural PHP, and therefore does not have classes. To compensate, the diagram simply represents the relationships between each PHP file in the program and any contained variables, functions and constants.



5.2.2 PHP File List

index.php

Login page, index page as it is the first page a user lands on when they access TaskerMAN. Not possible to proceed to any other file if they are unable to authenticate.

meta.php, footer.php, header.php

These files make up the static UI components, including the navigation and the HTML includes to any external stylesheets or JavaScript files.

init.php, config.php, connect.php

The three main supporting files of the software. Connect establishes and maintains a database connection to TaskerSRV. Config contains global configuration and settings. Init provides access to these resources as well as useful global functions throughout the entirety of the software.

taskerman.php, users.php

Task and User view of the TaskerMAN software respectively. The main files in the software that the user navigates to after authentication.

taskAdd.php, taskEdit.php, userAdd.php, userEdit.php, addElements.php, additionalElement.php

These are modal windows that are displayed on the Task or User view respectively. They provide functionality to add a task, edit an existing task, add a user, edit a user, add elements to a new task, and to add an additional element to an existing task.

addEntry.php, addUser.php, editTask.php, editUser.php

Carries out the adding or editing operations of a task or user, generating SQL and running them on TaskerSRV.

taskAddGateway.php

This file sanitises taskAdd input and prepares the data to be combined with the new task element data before posting it to addEntry. Since this is the most complicated operation, this file was needed.

taskDelete.php, userDelete.php

Deletes selected task or user. Will refuse to delete a user if they have tasks currently assigned.

addExtraElement.php

Adds an additional element to an existing task.

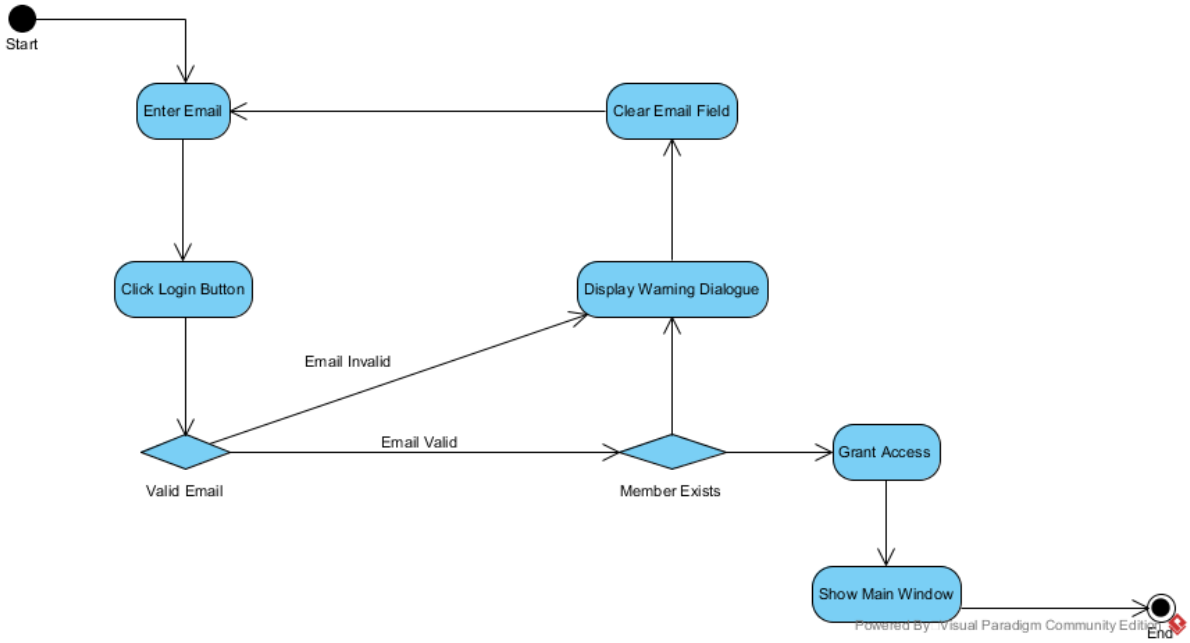
elementDelete.php

Deletes elements corresponding to a selected task.

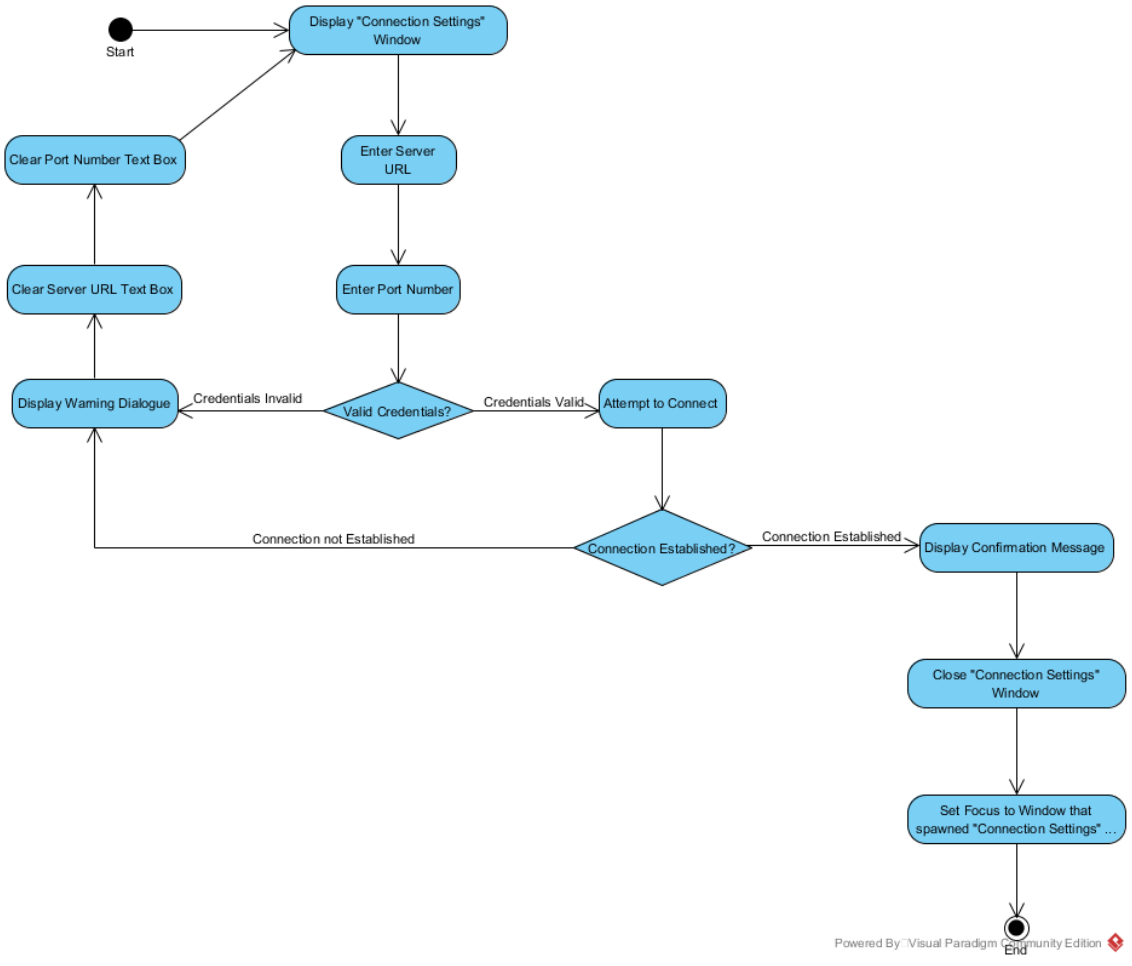
6. DETAILED DESIGN

6.1 Activity Diagrams

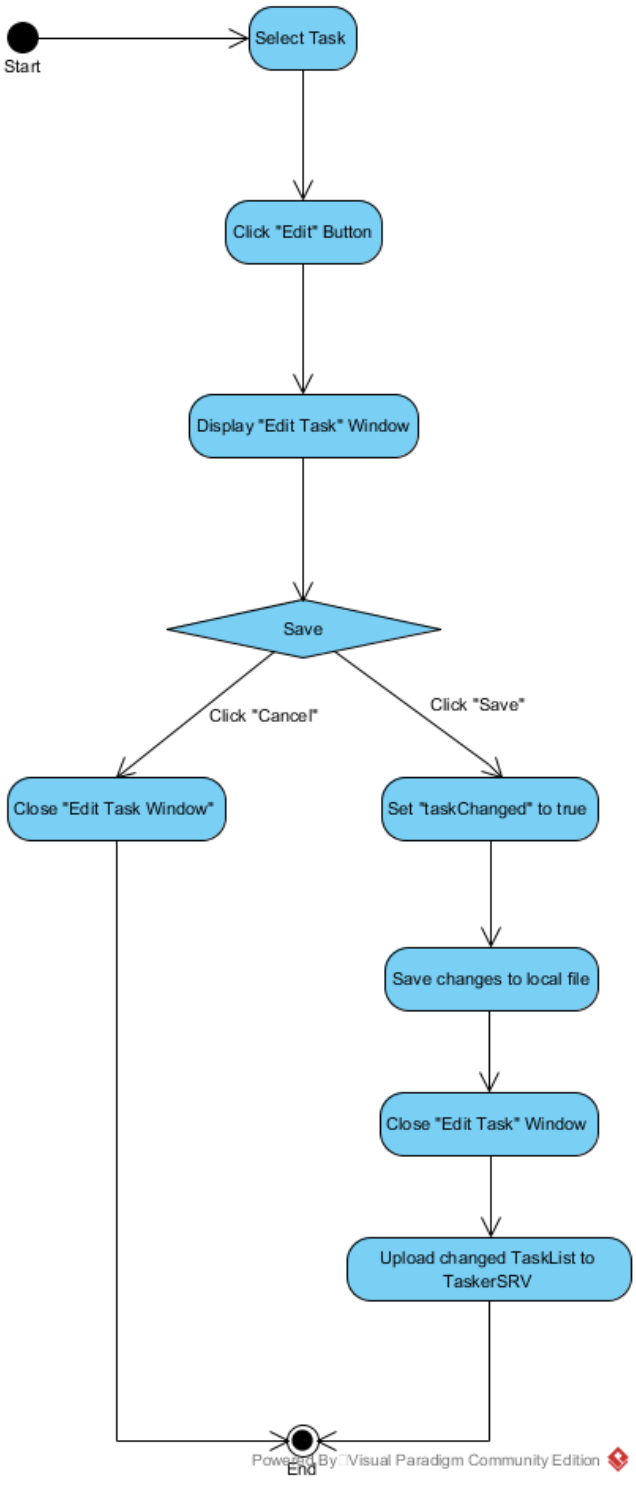
6.1.1 TaskerCLI User Login



6.1.2 TaskerCLI connecting to TaskerSRV

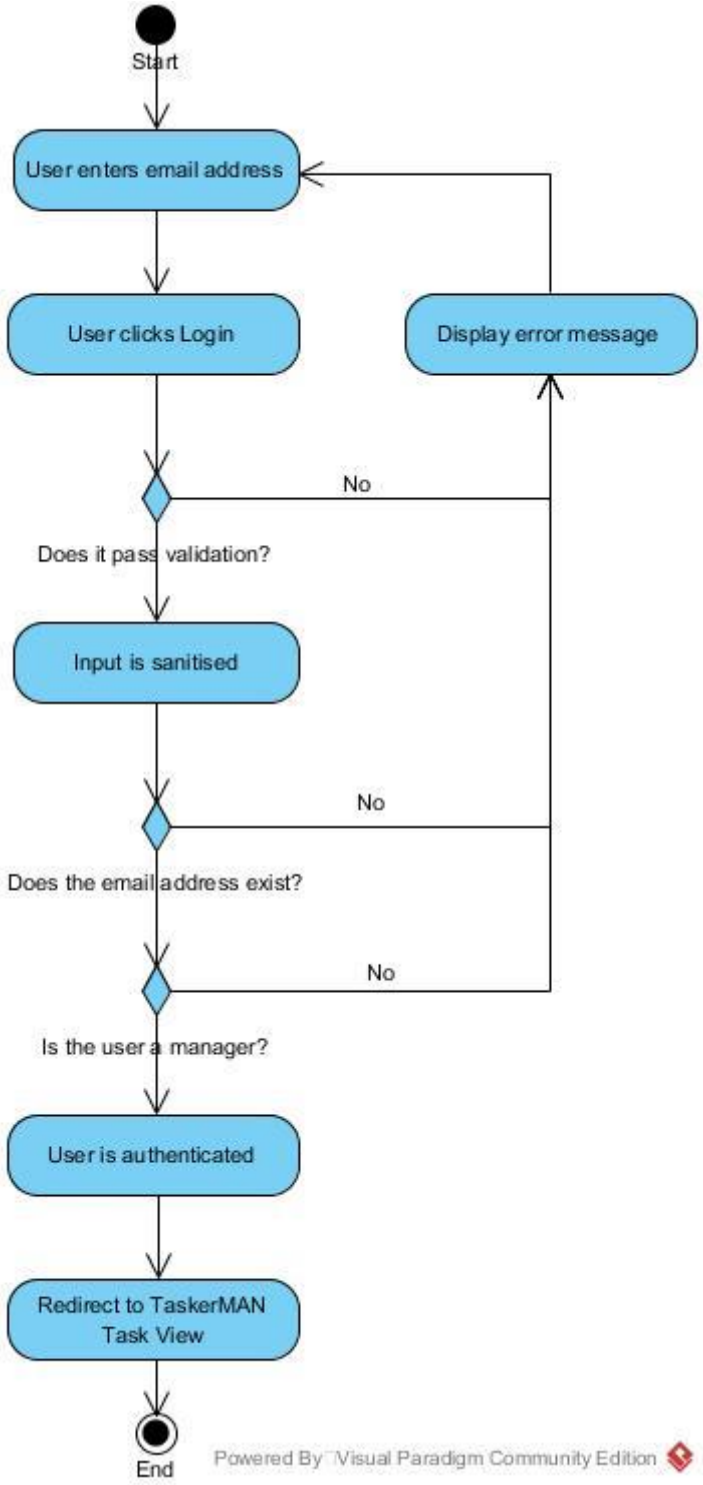


6.1.3 TaskerCLI Task Editing



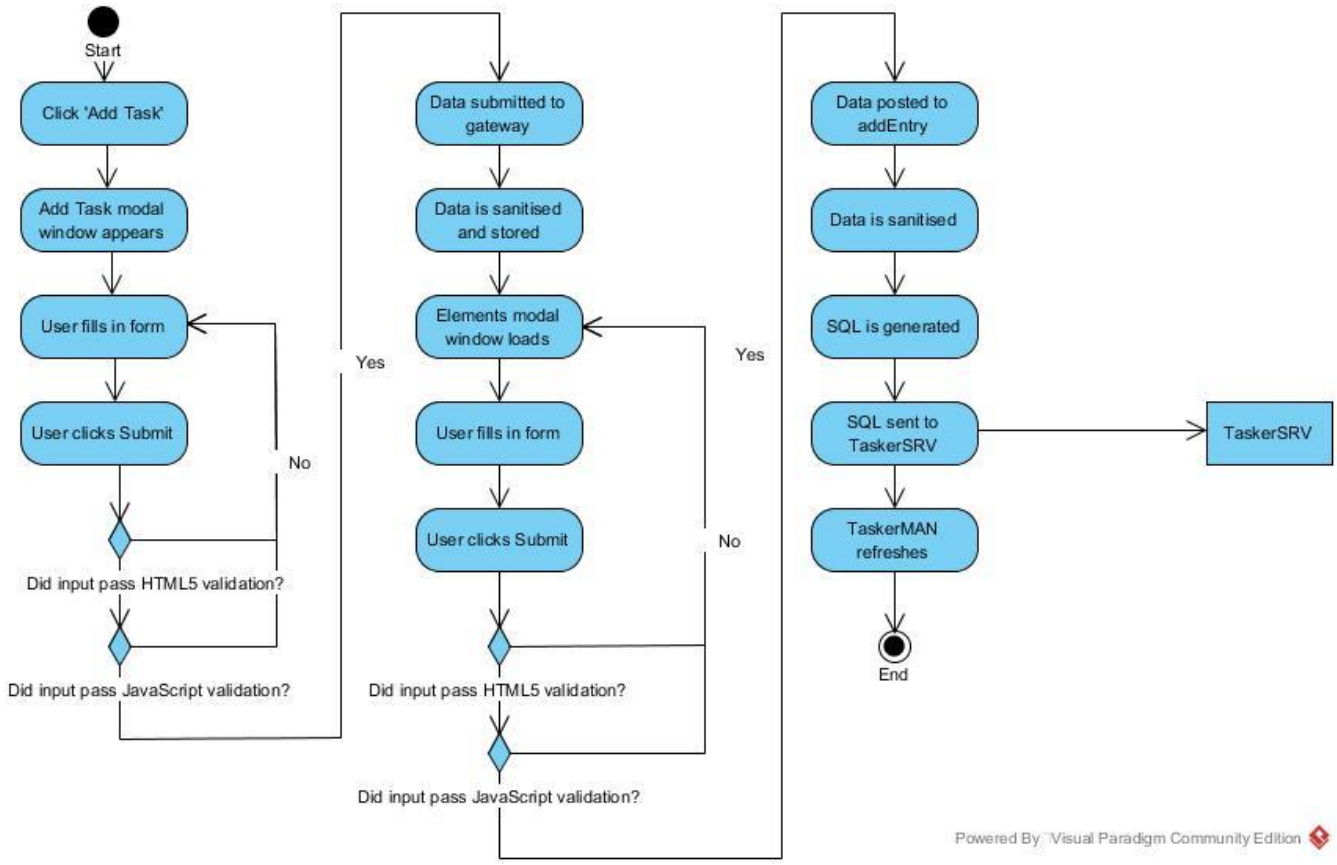
6.1.4 TaskerMAN Login

The login process for TaskerMAN was decomposed into discreet steps which are shown in the diagram below. This then displays the main window specified in FR7 to the user.



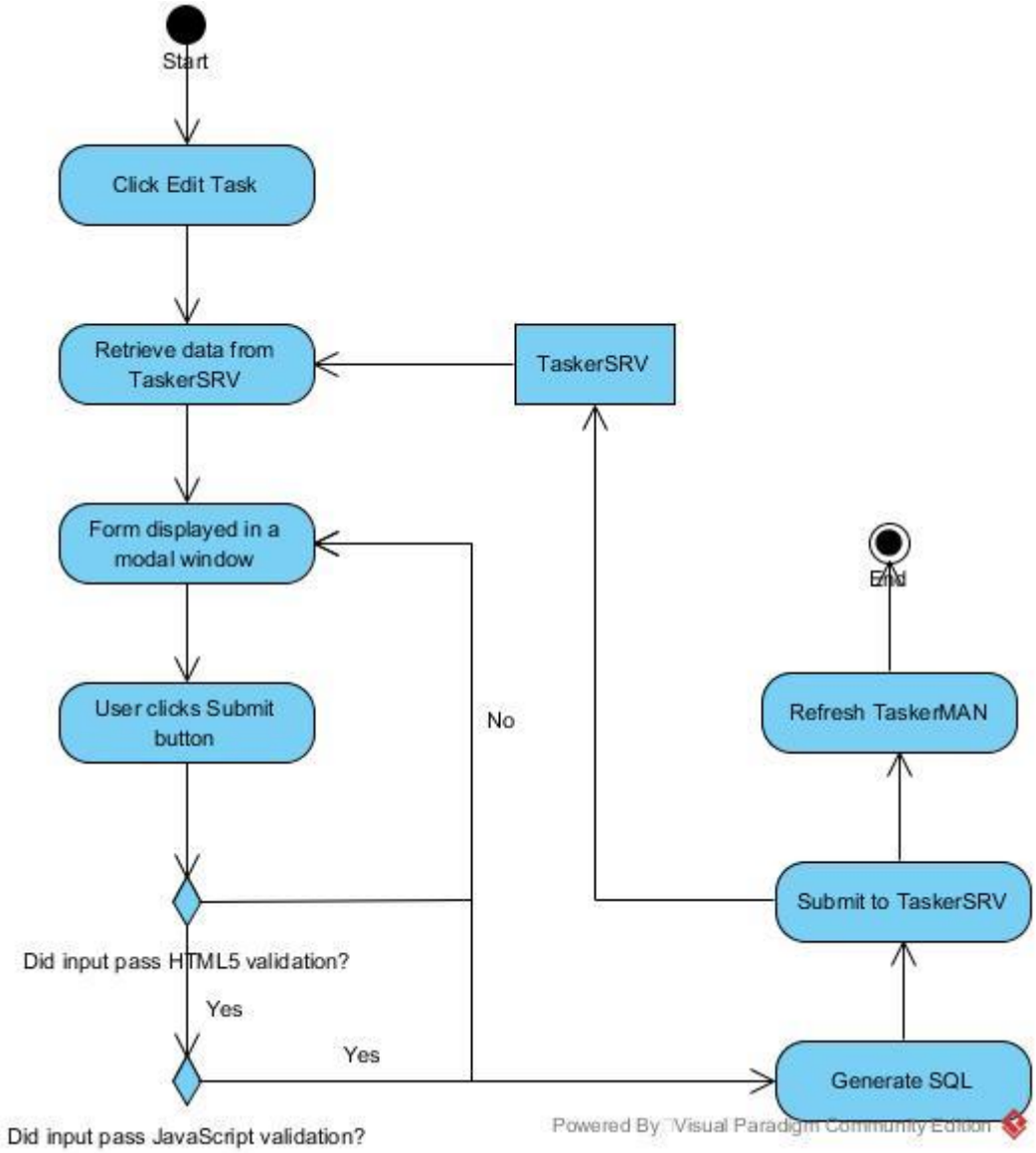
6.1.5 Adding Tasks in TaskerMAN

TaskerMAN must be able to add new tasks and task data to the system as per requirement FR4. This diagram below lists the activities which take place when a task is being added.



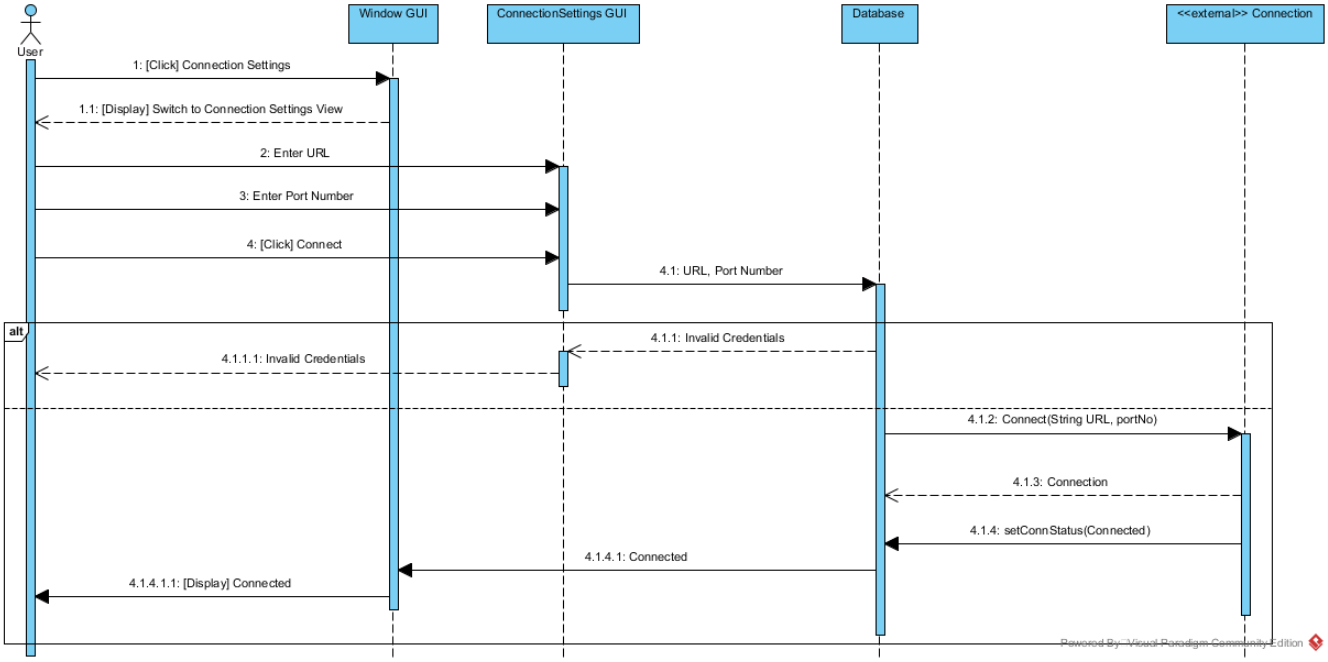
6.1.6 Editing Tasks in TaskerMAN

TaskerMAN must be able to edit tasks to allow reallocation or maintenance of task data as specified by requirement FR5. This diagram below lists the activities which take place when a task is being edited.

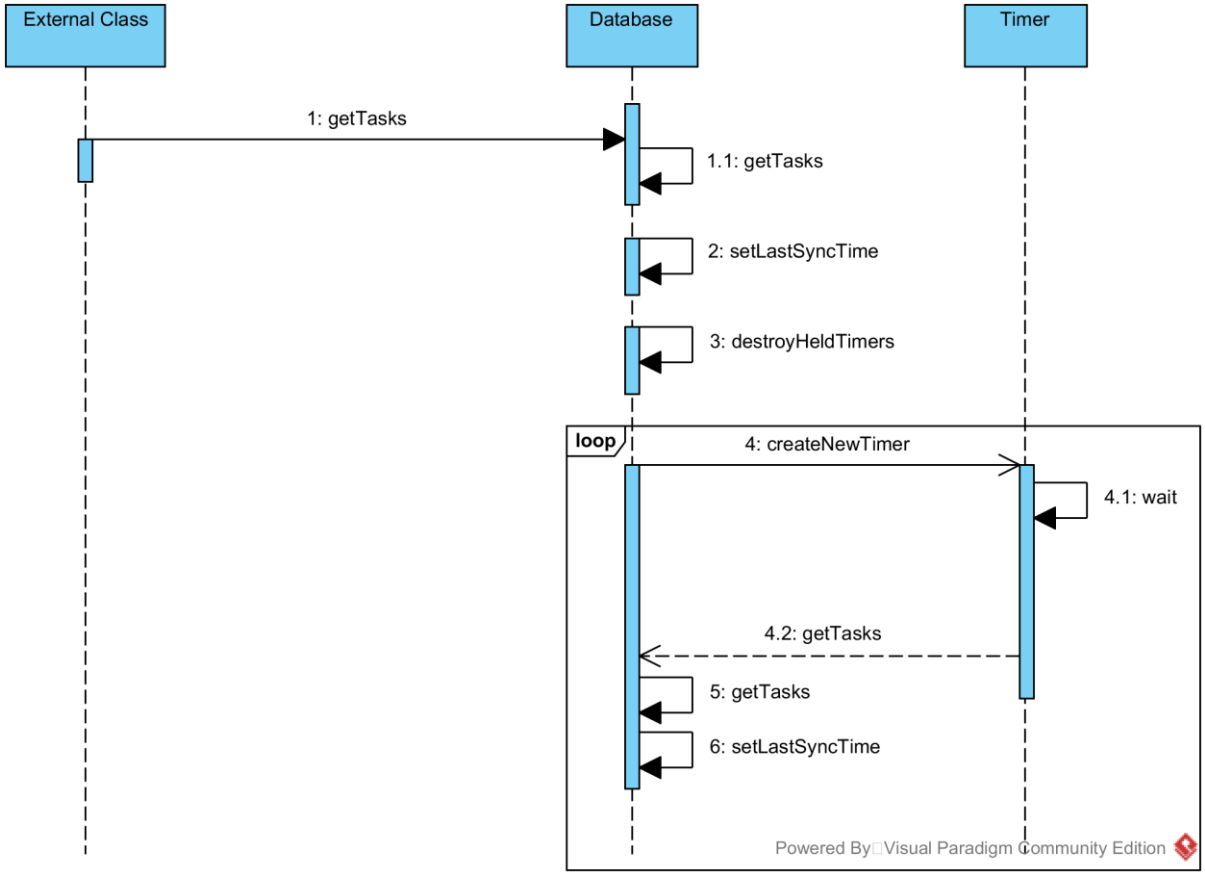


6.2 Sequence Diagrams

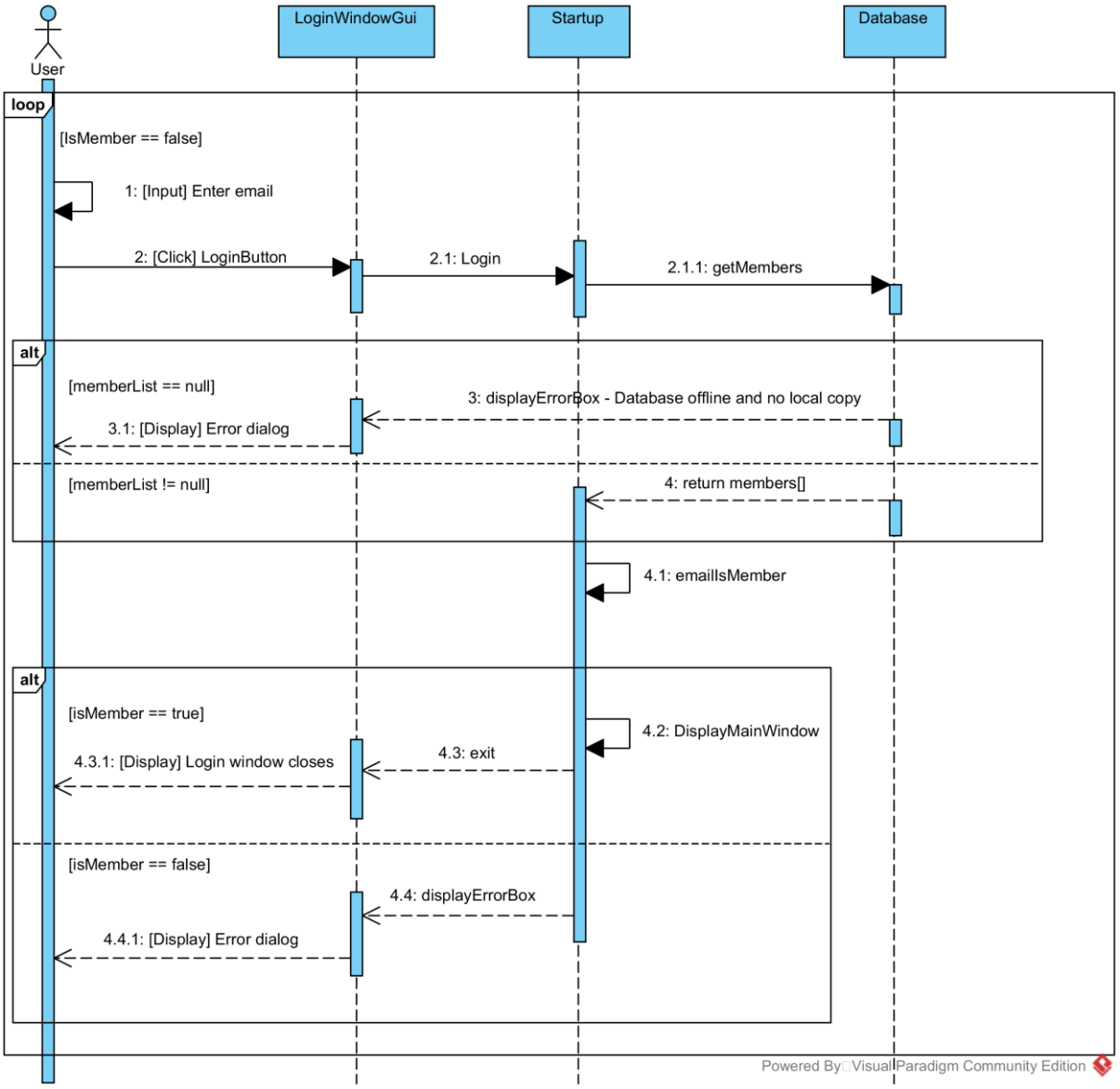
6.2.1 Connecting to TaskerSRV from TaskerCLI



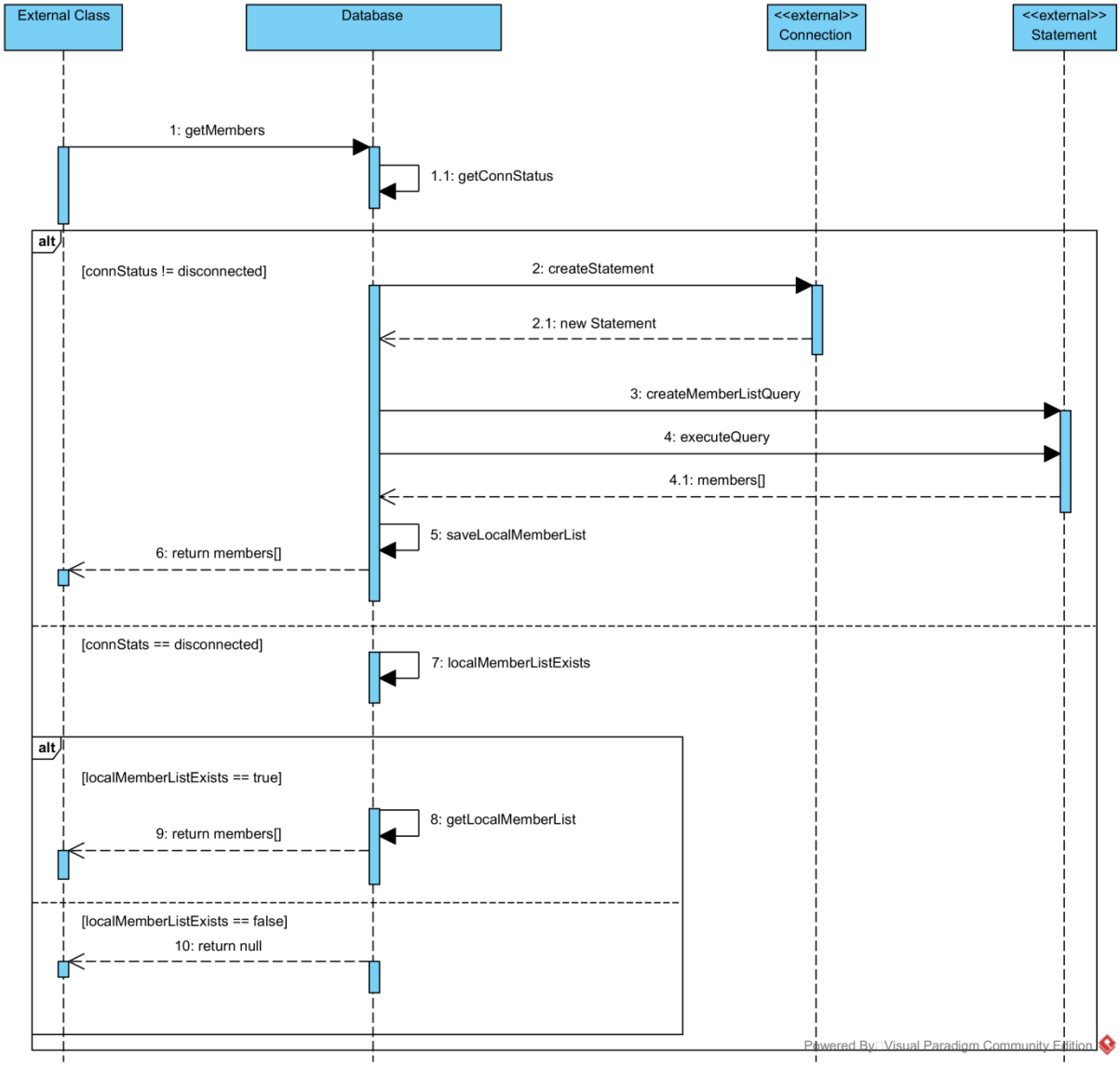
6.2.2 Automatic synchronization to TaskerSRV from TaskerCLI



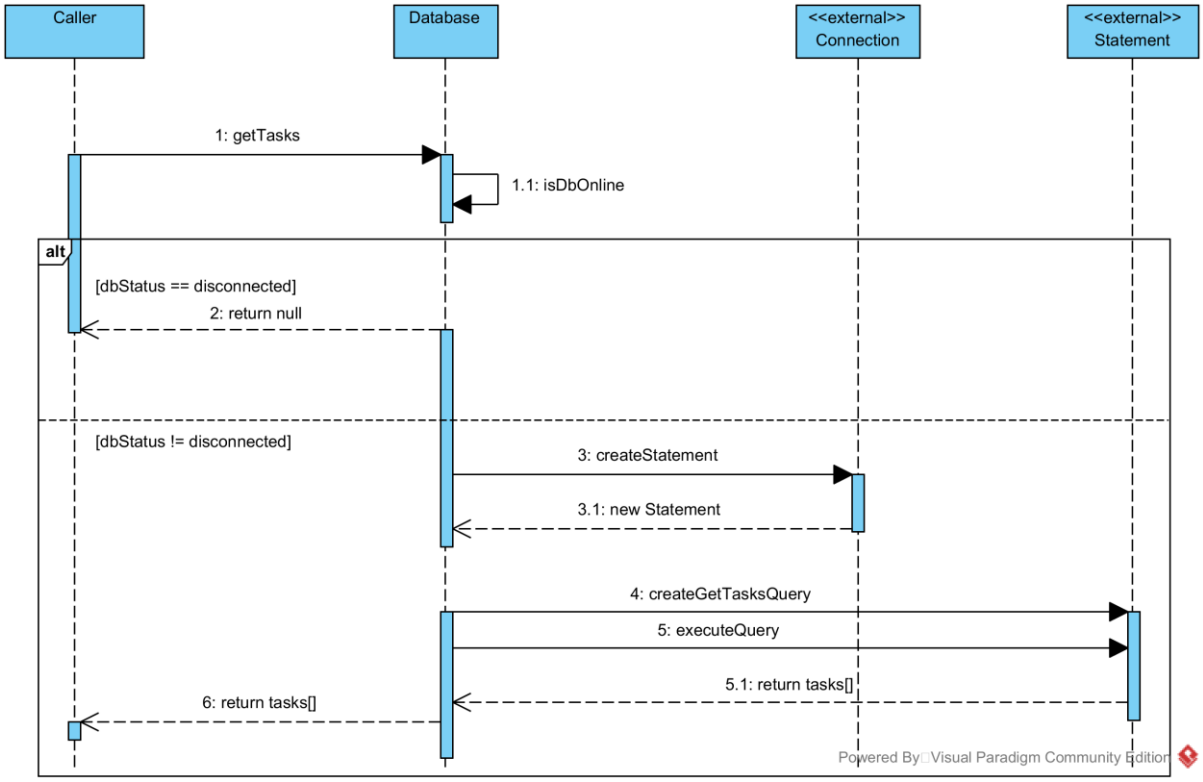
6.2.3 User Login to TaskerCLI



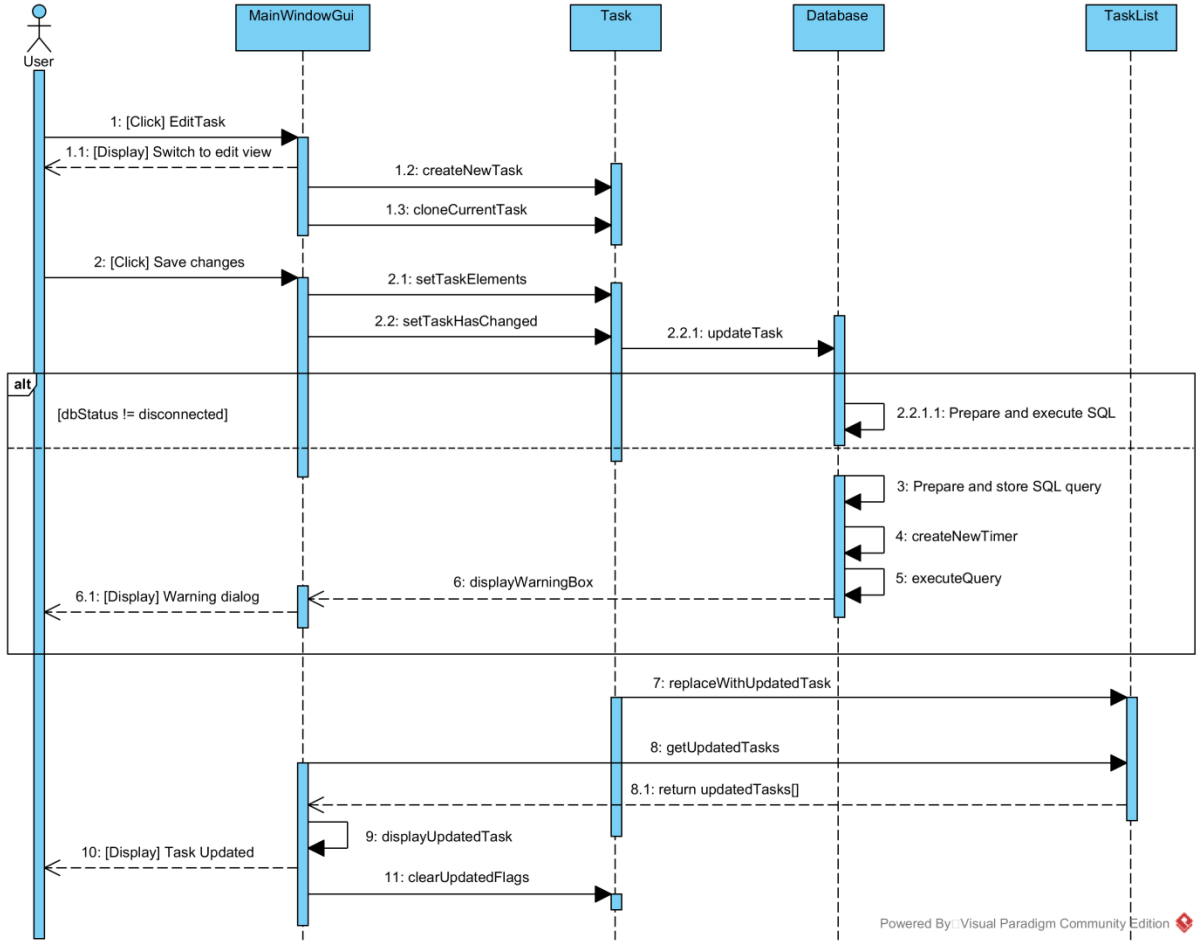
6.2.4 Get list of members in TaskerCLI from TaskerSRV



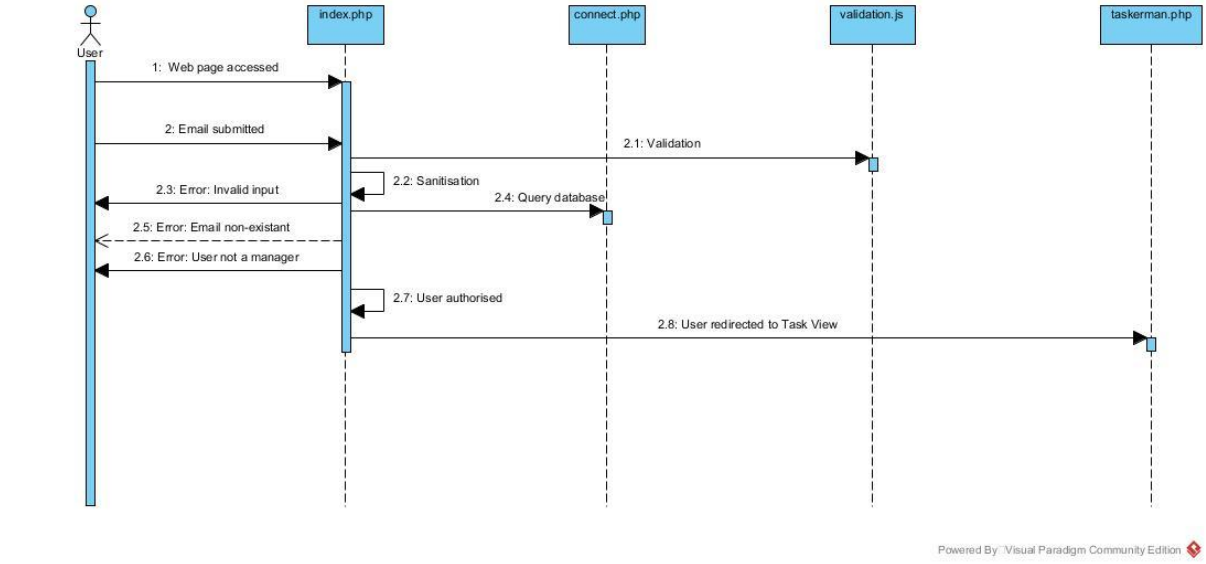
6.2.5 Get list of tasks in TaskerCLI from TaskerSRV



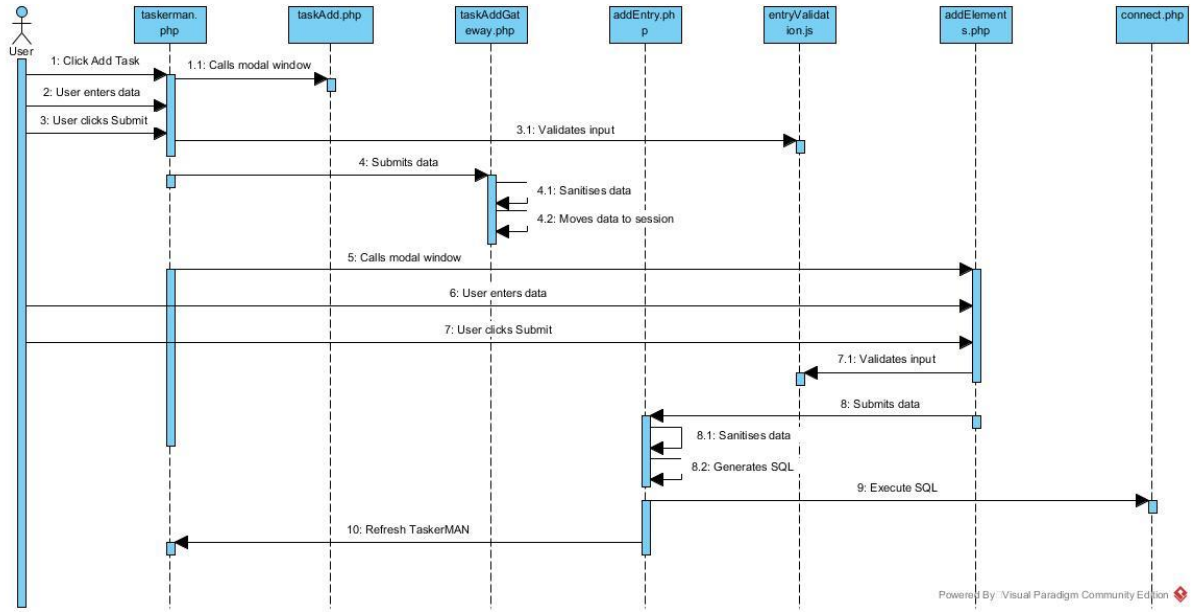
6.2.6 Editing tasks in TaskerCLI



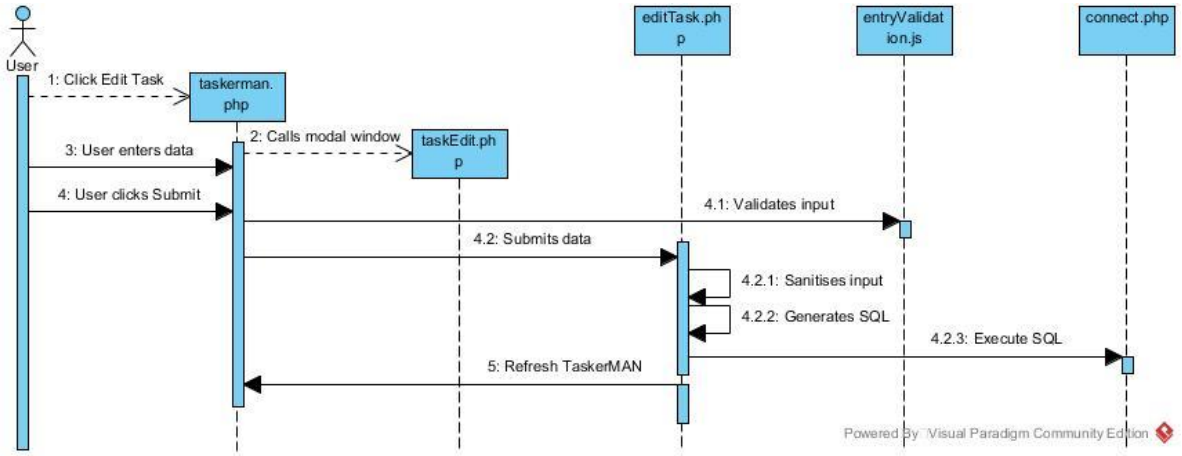
6.2.7 User Login in TaskerMAN



6.2.8 Adding Tasks in TaskerMAN



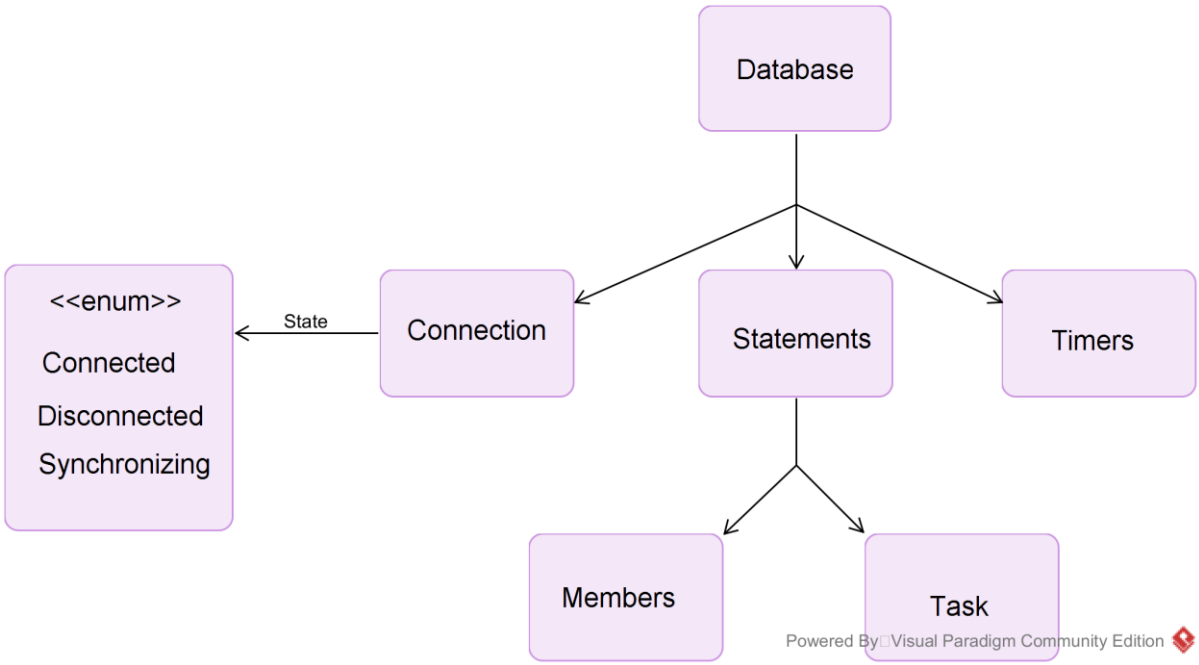
6.2.9 Editing Tasks in TaskerMAN



6.3 TaskerCLI Data Structures

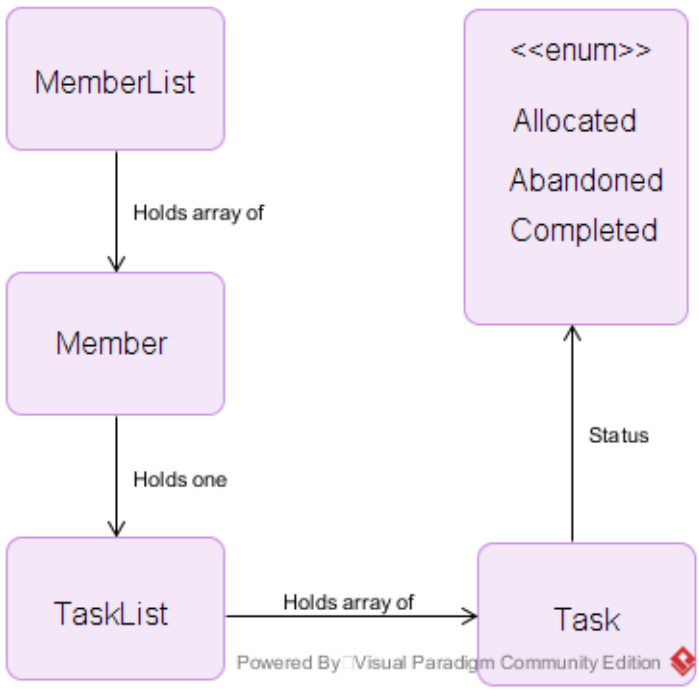
6.3.1.1 Database Structure

The diagram below shows the data classes the Database class uses in order to function



6.3.2 Members and Tasks Structure

The below diagram shows the hierarchy of Member related classes and Task related classes. From this diagram it is obvious the number and type of objects a class will hold or use.



6.4 Spike Work

6.4.1 TaskerCLI

The largest risk to TaskerCLI was implementing GUI's. This was identified as a significant risk early in planning so spike work was created to assess and mitigate any issues. For this spike work several people created the mock ups in Java using various tools.

Window builder allowed us to rapidly and accurately create the required design for the GUI; however the designer can be unintuitive with minor changes requiring major layout method changes. The group met and discussed finding and identified common pitfalls and how to avoid them, such as using several frames where layout could change.

In the process we established that by using an interface on all window classes we could easily construct and destruct these windows whilst abstracting over the minor differences between them.

This concept leads to creating a window manager which handles opening and closing windows on behalf of the software. Refining the code the final design was an array of window interface objects. By using an enumerated list we can access windows and manipulate them by doing a single line such as *"windowManager.createWindow("MAIN_WINDOW_ENUM");"*

6.4.2 TaskerMAN

Once PHP was chosen to drive TaskerMAN the team looked into PHP Unit. Having all used and understood the Java equivalent Junit the group needed to check if this knowledge was applicable to PHP Unit for creating unit tests.

This spike work consisted of setting up and learning how to implement PHP unit tests. One outcome which came to light was that our planned usage for PHP is mostly procedural whilst PHP unit caters for object oriented paradigms. The group also had to train each other in configuring the IDE to correctly implement these tests and writing suitable tests in PHP Unit.

Additional spike work on input validation was performed for TaskerMAN. As emails are used as an input we needed to check the input before sending it to the server. Several approaches were discussed such as using HTML 5 and JavaScript tests to detect invalid input on the client side. The approach decided upon was regular expression tests to ensure input is valid.

REFERENCES

[1] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

[2] Tutorialspoint, "TutorialsPoint JUnit Environment Setup," TutorialsPoint, 27 10 2015. [Online]. Available: http://www.tutorialspoint.com/junit/junit_environment_setup.htm. [Accessed 27 10 2015].

[3] Earl, Oliver; The PHP Group, "PHPInfo running on Apache," 27 10 2015. [Online]. Available: http://users.aber.ac.uk/ole4/phpinfo.php. [Accessed 28 10 2015].

[4] S. Bergmann, "PHPUnit English Documentation," 28 10 2015. [Online]. Available: https://phpunit.de/manual/current/en/phpunit-book.html#installation.requirements. [Accessed 28 10 2015].

[5] Oracle, "Oracle Software Delivery Cloud - MySQL Standard Edition for Linux x86-64," Oracle, 2015. [Online]. Available: https://edelivery.oracle.com/osdc/faces/SearchSoftware?_adf.ctrl-state=nmw6458k7_28&_afrLoop=2752661125326430. [Accessed 21 10 2015].

[6] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.05 A 1.8 - Design Specification Standards,* Aberystwyth University: Software Engineering Group Project, 2015.

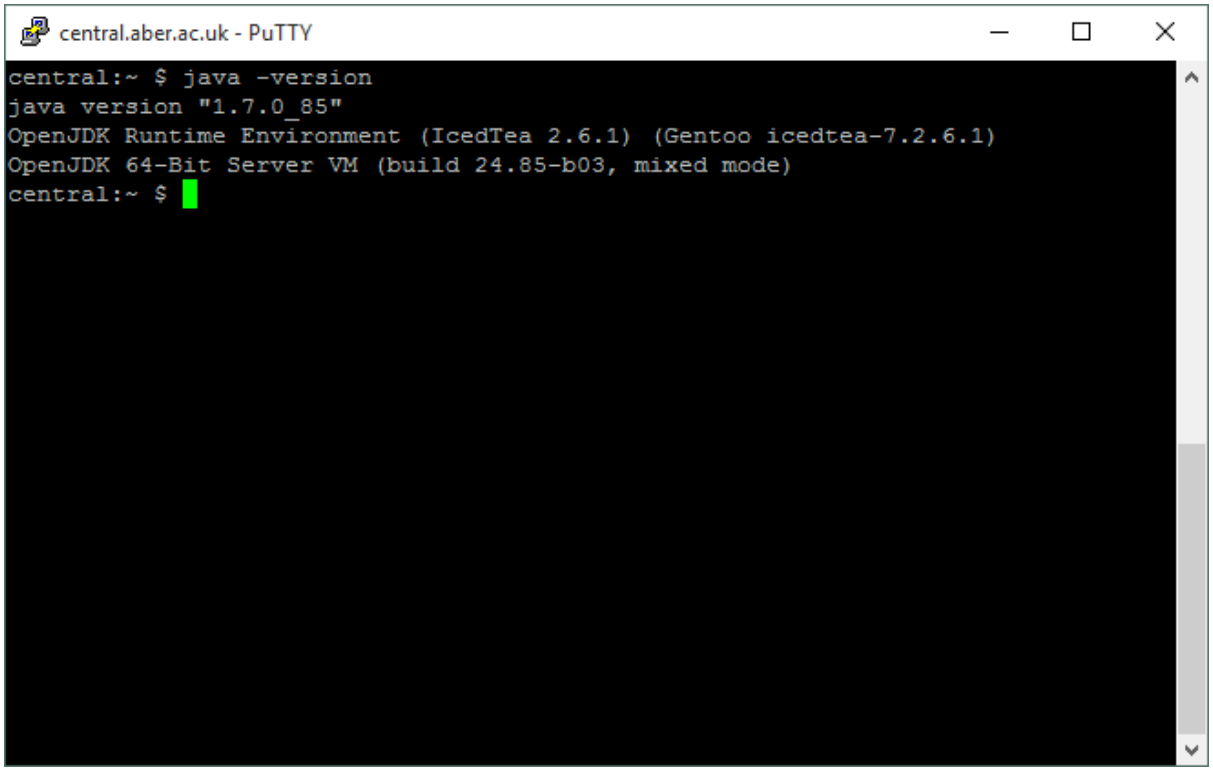
[7] Pixeden, "Firefox Web Browser Mockup Template," CorruptedDevelopment, 26 08 2014. [Online]. Available: http://corrupteddevelopment.com/firefox-web-browser-mockup-template/. [Accessed 26 10 2015].

DOCUMENT HISTORY

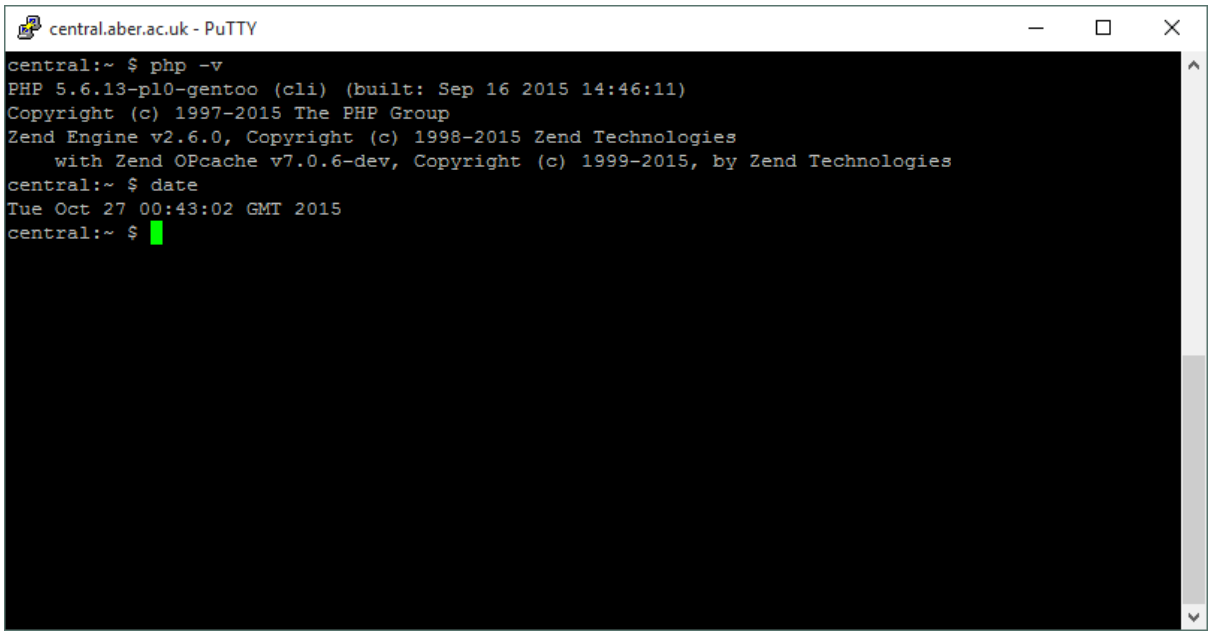
Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	27/11/15	Original version	DAF5
1.11	34	14/12/15	Changed interaction diagram to correctly show PDO instead of MYSQL	DAF5
1.2	48	14/12/15	Updated TaskerCLI class diagram with correction	DAF5
2.0	187	13/02/16	Major revision to design documentation following multiple changes throughout implementation of software.	DAF5

APPENDICES

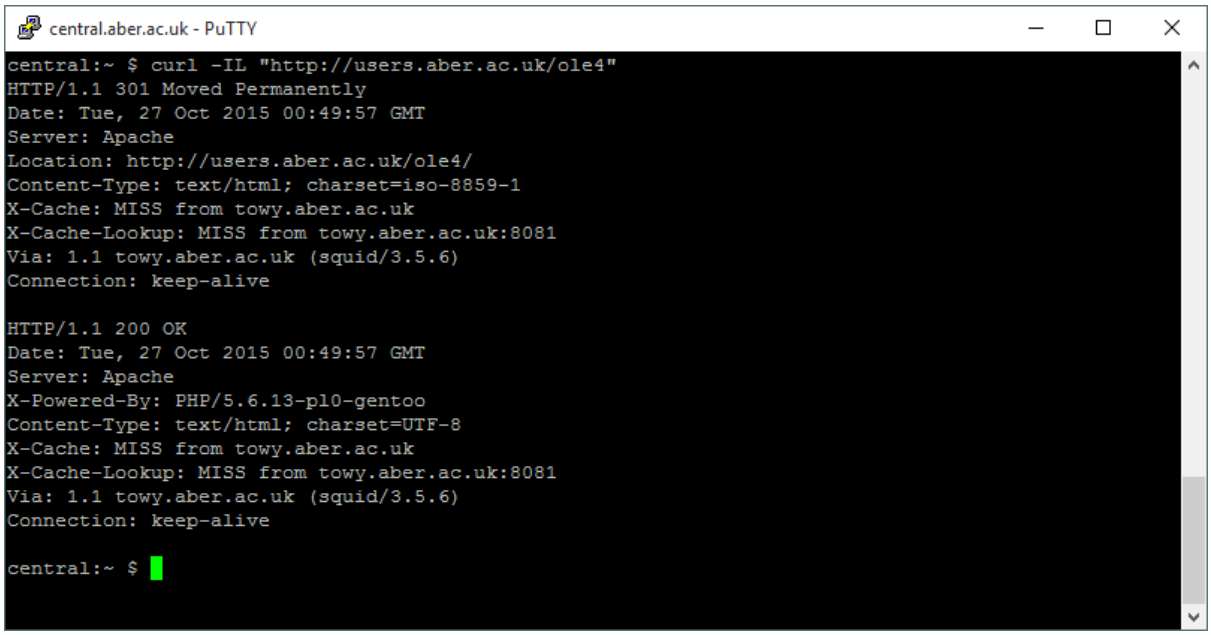
APPENDIX A – Java Version



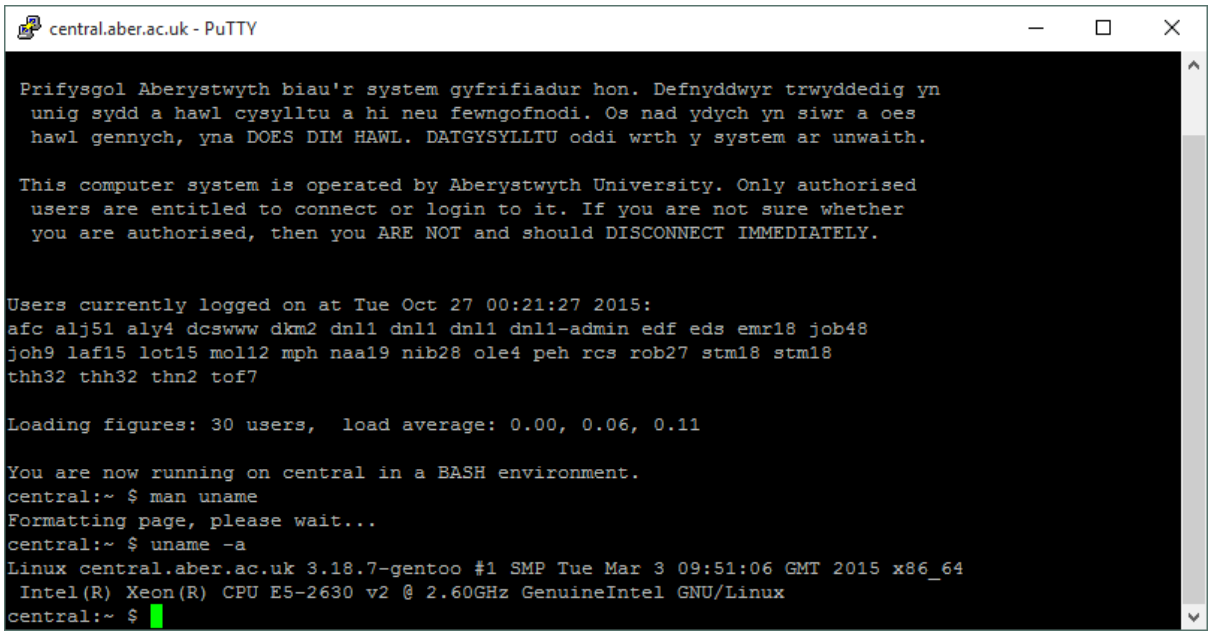
APPENDIX B – PHP Version



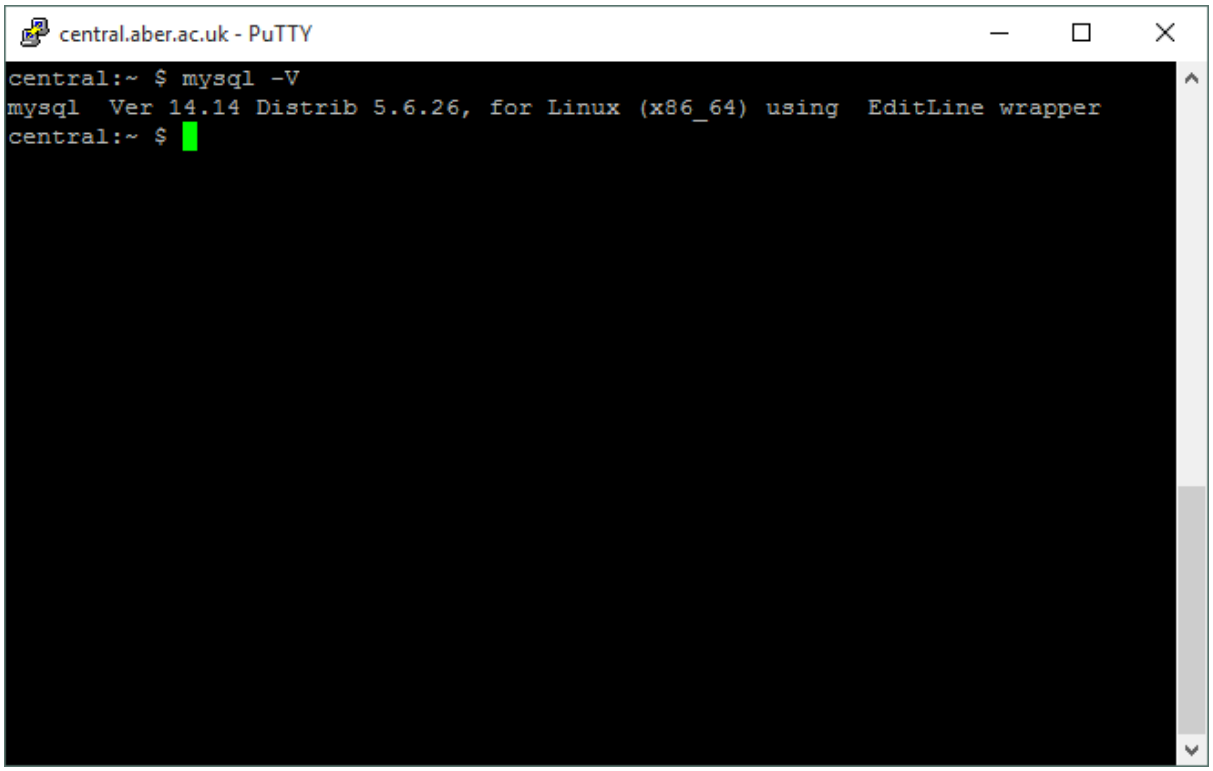
APPENDIX C – Apache Information



APPENDIX D – Linux information



APPENDIX E – MySQL Version



APPENDIX F – Logic Class Diagram