Table of Contents

1. **Background** 1
2. **Team organisation** 1
3. **Managing the software development process** 5
4. **My role in the team** 9
5. **Quality Criteria** 11
6. **Recommendations for undertaking similar team projects in the future.** 14
7. **References** 17
8. **Appendix – Gantt Chart for Team 23** 18

Developing Quality Software: Individual Report

1. Background

This report aims to evaluate the techniques used and reflect on learning gained during my experiences whilst working as part of a team whilst developing a prototype personal and academic tutor management system (TMS) for COMSCI. This team exercise required us to develop the TMS software system for use by the school's senior personal and academic tutors. The first part of Coursework 1 (Part A) was to ensure the system met a set of functional and non-functional requirements and to develop a project plan with a Gantt Chart showing scheduled timings for all key tasks. We also had to develop a risk assessment plan for the project. The second part of the exercise (Part B) was to develop a design model using object orientated design for the TMS based on Part A. Coursework 2 was to develop a working prototype of the TMS using *Python GUI* Programming (*Tkinter* module), covering all major functional requirements and using a set of test cases to ensure validity.

2. Team organisation

2.1 Tuckman's Theory Forming, Storming, Norming, Performing

In lectures, I had learned about how a group of people can develop into an effective team with a shared vision quicker if they understand theories of team dynamics. One very useful theory is Tuckman's teamwork theory (Tuckman, 1965), which describes the various stages a group passes through on its way to becoming an effective team. This model has five stages: Forming, Storming, Norming, Performing and Adjourning as shown in Figure 1.

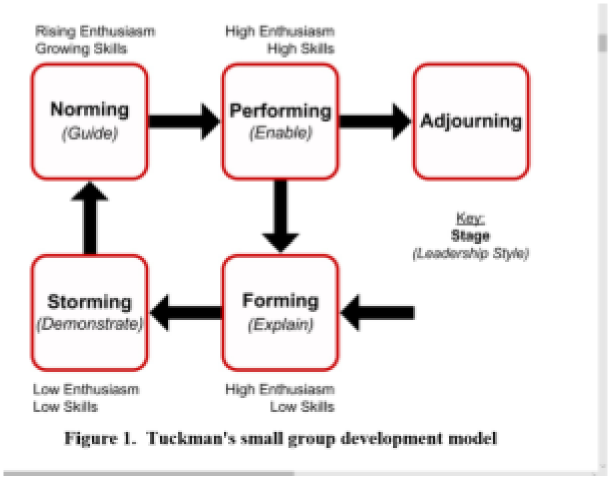


Figure 1 Tuckman's small group development model (from Largent and Luer, 2010)

According to Figure 1 the stage a team falls into can be determined by the combination of their enthusiasm and skill levels. However, these five stages can also be described in terms of effectiveness, which takes account of possible challenges and difficulties which arise during the project as shown in Figure 2.

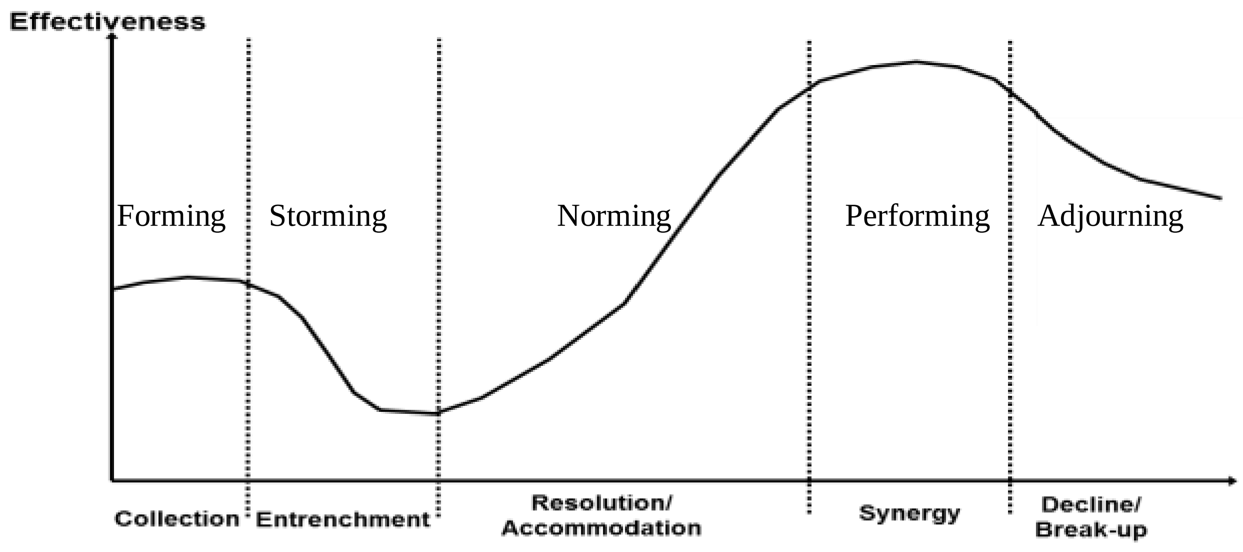
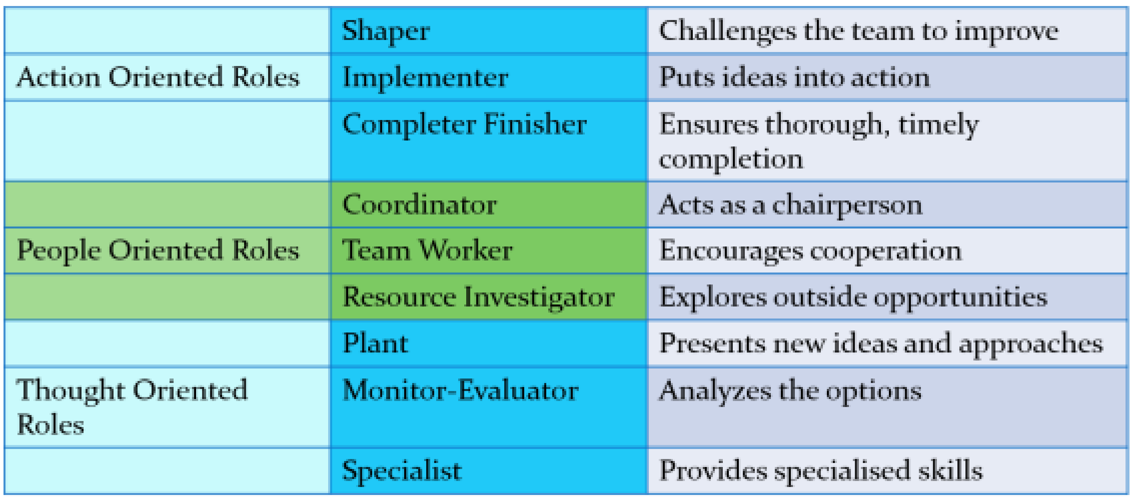


Figure 2 The Lifecycle of Teams: Effectiveness profile of team lifecycle (from CM1202 lecture)

<u>2.2 Belbin's theory</u>

Another theory which was discussed in lectures was Belbin's team role model (Belbin, 1981). Belbin suggested that people on teams assume different "team roles" and described how a "team role" is a tendency to behave, contribute and interrelate with others in a certain way. Such roles will ultimately determine the team's success. The theory is based on nine team roles categorised into three groups: Action Oriented (shaper, implementer, completer/finisher), People Oriented (Co-ordinator, team worker, resource investigator), Thought Oriented (innovator, monitor evaluator, specialist) (Table 1).



	Shaper	Challenges the team to improve
Action Oriented Roles	Implementer	Puts ideas into action
	Completer Finisher	Ensures thorough, timely completion
	Coordinator	Acts as a chairperson
People Oriented Roles	Team Worker	Encourages cooperation
	Resource Investigator	Explores outside opportunities
	Plant	Presents new ideas and approaches
Thought Oriented Roles	Monitor-Evaluator	Analyzes the options
	Specialist	Provides specialised skills

Table 1 Belbin's Team Roles Table (from CM1202 lecture)

2.3 Tuckman and Belbin theories applied to practice

Looking back on my experiences I can see how both Tuckman's teamwork theory and Belbin's team roles theory can be useful in helping develop an efficient and effective team, as will now be shown. In Tuckman's 'forming stage' (Figure 1) individuals first come together, this is also called 'collection' (Figure 2). At this stage people have high enthusiasm but low skills. This was the case when my group first formed. Thus, following receipt of the students' names for our group, the four of us who were already friends set up a group chat on Facebook to help with communication. We E-mailed the other two students but did not get a reply. Eventually we discovered that they had left the course, which meant we only had a team of 4 people. Nevertheless, we were excited and enthusiastic even though we knew that our skills sets were low and we had a great deal to learn if we were to ensure success of our project. Forming is also when members establish themselves. For my particular group this was easy as we had all worked together before, had a good rapport and already had knowledge of the relative strengths and weaknesses of the group. This gave us a head start since Belbin highlights that having such knowledge can help manage interpersonal differences in a team and allow identification of the relative roles each person can take, making sure the team is balanced.

During the 'forming stage' members must also establish what is expected of them. Therefore, following brainstorming, we discussed ideas and established that the best way forward was to distribute the tasks equally between us all. After further discussions, we finally decided on what we each felt confident in doing and the various team roles were assigned. I was assigned to code the first part of the application, opening the CSV (comma separated values) file, uploading the information into the systems database and assigning students to tutors. We then agreed to work on our individual tasks whilst keeping in touch through the group chat.

The 'storming stage' is when enthusiasm and skills are low (Figure 1). Generally, very little productive work gets accomplished during this stage. Conflict can sometimes arise because individuals do not accept the group's decisions and have preconceived ideas as how the project should proceed. We experienced this unproductive phase. A major challenge initially was that we had less people than most other groups thus individually had less expertise to draw upon. We started to feel daunted by the challenge of completing the volume of work in the time available. Also, there was some conflict over our tasks and responsibilities, and sometimes we became frustrated as one person was waiting on completed work from another person. Some people were also rather defensive, explaining how the huge volume of other coursework to complete was competing for their time. It was clear that we were finding it difficult to deliver our individual pieces of work as originally planned. We therefore decided to regularly get together to work collectively. At this stage things started to improve. We were clearly entering the ‘norming stage’ (Figure 1) also known as the resolution stage where effectiveness increases (Figure 2). We started to go into university to work together and share ideas as a group. Sitting together and working collaboratively while we were next to each other helped enormously. It made many of the tasks a lot easier as we could bounce ideas off one another and resolve any disagreements quickly and collaboratively. Often two people were working on the same thing to help with efficiency. During this time, we successfully developed our strengths and managed our weaknesses as individuals within the team. We were also flexible which meant our allocated tasks could change or overlap. For example, in addition to the original coding allocated to me I agreed to integrate the final code and where necessary help debug the final TMS prototype.

These were also examples of Belbin’s team theory, where for success there needs to be a balanced team. According to this theory, teams can become unbalanced if all team members have similar styles of behaviour or team roles. Thus, by collaboratively working together through direct contact we could recognise strengths and shift out roles to “balance” our team. For example, one of my weaknesses was documentation, so I agreed to do more of the debugging and integration whilst others could do more of the testing and complete the necessary validation documentation. These experiences align with the roles within Belin’s theory shown in Table 1. Thus, looking back, I believe at the start of the project, two of the team were action orientated, one people oriented and the fourth thought oriented. Thus, one of the team was a ‘completer finisher’ driving the project forward by ensuring thorough and timely completion, another was an ‘implementer’ putting ideas into action. Meanwhile another member tended to encourage cooperation and I was the person who presented new ideas. As the project proceeded and our skill sets and confidence expanded so did our roles. Thus, I started to provide specialist skills in the coding, whilst another member assumed a role of chairperson, helping to coordinate the work. This helped further balance the team. As we progressed, our confidence and trust in one another increased as did our self-esteem. Also, as disagreements got resolved and as productivity improved, our enthusiasm started rising and our skills improving. These are all typical of the ‘norming stage’.

As the deadline of the project approached, we really pulled together and worked hard to meet the project requirements. This represented the ‘performing’ stage where skills and enthusiasm were high (Figure 1). At last the system we were creating was beginning to make sense. We had begun to work a lot faster and had clearly peaked in effectiveness, also synergy within the group was obvious (Figure 2). At this point we really had developed into a proper team, with team performance being more than the sum of its parts. Eventually we got to the stage where we were happy with

Coursework 1 and one member of the group was tasked with submitting it. Unfortunately, with Coursework 2, a whole new skill set was required and we came across coding difficulties when trying to get the prototype TMS to work. This set us back and we essentially returned to the low productivity/effectiveness of the storming and norming stages for quite a while. When the deadline of the project approached for Coursework 2, we really pulled together and gave 100%, working long hours to try to meet the brief. Unfortunately, as the deadline was upon us, we had to resign ourselves to not being able to deliver on all of the requirements and because of the lack of time we failed to complete all the testing and documentation for the test cases.

2.4 Barriers

The biggest barrier was developing the coding skills to successfully complete the requirements of Coursework 2 efficiently and effectively. This was reflected by our inefficiency and spending too long in the 'storming phase'. Another major barrier was time management and was reflected by our Gantt chart where we failed to include sufficient time for the implementation and testing of the TMS.

None of our group had experience in coding prior to embarking on the COMSCI course so were all novice coders. This meant we did not have a definite 'specialist' in the team at the start. Whilst we were trying to work together and work on the coding in pairs to overcome the lack of technical expertise, it became evident that one person was adept at putting forward ideas and driving the forward the coding and implementation. Therefore, as the project progressed, we adjusted people's roles so they could contribute to more certain in areas. For example, although I was initially tasked with the initial code to upload the CSV file, I also helped debug other parts of the system code. For both sets of coursework we pulled together and worked hard right to the very last minute, but we were let down by our poor time management. Ultimately, we did manage to deliver on most of the requirements, as reflected by our 'excellent' for validation. However, although our system worked, the clarity of instructions, relevance and the documentation which covered the test cases were of poor quality. This is reflected in our low mark.

3. Managing the software development process

The TMS prototype was developed using a waterfall approach (van Vliet, 2008) with requirements, design, implementation and testing following on one after the other. In following this approach, we needed to manage the project, come up with version control strategies and evaluate risks. Here I will evaluate my team's performance during the project, based on these requirements.

3.1 Project Management

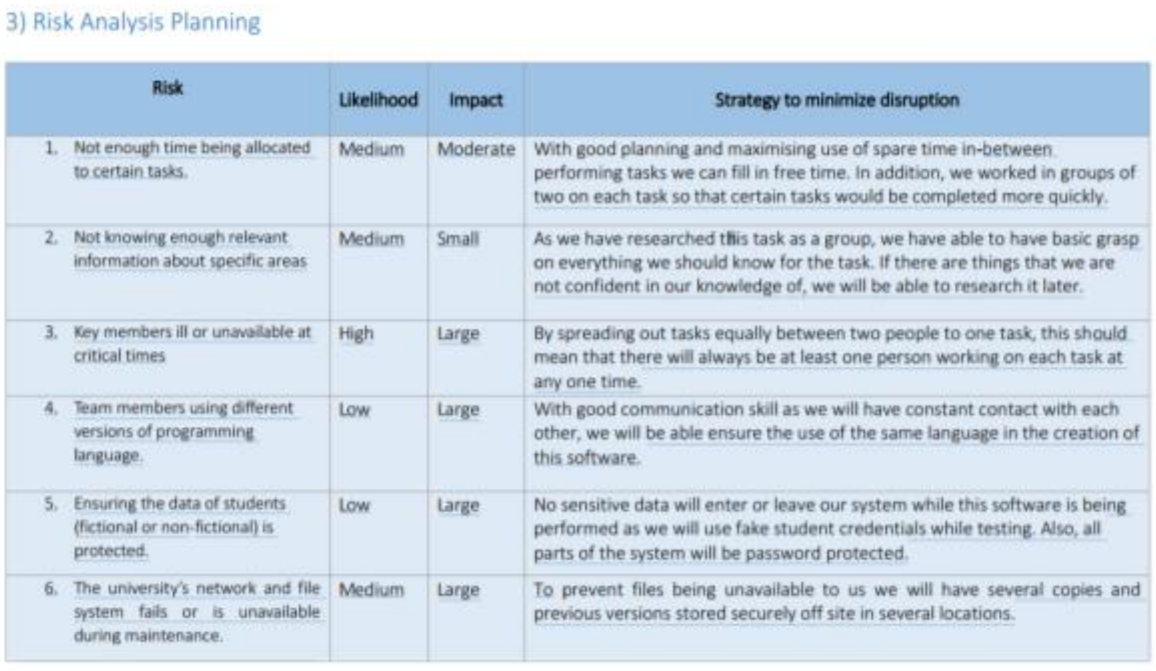
To manage the project effectively and ensure we met the requirements within the timeframe allocated, good planning was needed. This meant breaking the project down into manageable sized bits, allocating them to the team members and scheduling them. Firstly, we needed to agree the main functionality required of the system and identify what architecture and data structures we needed for the system to work properly. We decided to simply break down the various parts of the project into manageable areas. For example, we needed to determine functional requirements based on MoSCoW notation and non-functional requirements with acceptance and test criteria, develop a use case model and UML (Unified Modelling Language) class diagram, determine risk analysis and design a Gantt chart.

We started by getting together to brain storm and document all the functional and non-functional requirements that could be implemented in our system. This meant that we could build an efficient system that had just the right number of features to do its job effectively. The next step was to allocate tasks and plan how we were going to use our time to complete each aspects of our system. We decided not to assign a project manager as we thought we were familiar with each other's capabilities and could work well as a team with communication via group chat. However, I did help initially to assign a lot of the tasks to the team members and provided ideas to help drive the project forward. After group consultation, the tasks were allocated evenly across all four team members, based on each person's perceived individual skills. The most effective way to schedule the work was to design a Gantt chart. We found it challenging when we had to estimate the length of time it would take to complete each task. We also had to decide whether there were any dependencies. This approach would help us to use time effectively and facilitate effective planning of the project. Then one person was tasked with creating a professional looking and well-designed Gantt chart, with dates for key deliverables and milestones (see Appendix).

Although we produced an excellent, professional looking Gantt chart, we omitted to include the time for coding, implementation and testing of the system. In fact, time management of this part of the coursework posed quite a problem for us, and we ran out of time. We had underestimated the length of time it took to produce a complete prototype of the TMS. Since the testing was dependent on this, this part of the task suffered significantly. A further project management issues was that in our plan we failed to properly understand some of the work detail needed. For example, for the non-functional requirements we failed to write acceptance criteria which were measurable using appropriate metrics and we did not justify our design and suggest alternative flows. These problems also affected the mark we had.

3.2 Risk Management

Risk management is vital for the success of a software development project. It “concerns the identification of potential problems, so that timely actions can be taken to mitigate adverse effects” (van Vliet, 2008). To achieve adequate risk management, we needed to develop strategies to manage risks including avoidance strategies, minimization strategies and contingency plans. We also needed to consider actions to minimise disruption and regarding information to collect whilst monitoring the project to anticipate problems. The team therefore identified 6 risks following brainstorming, estimated their likelihood and impact and suggested various strategies to minimise their disruption (see Table 2). We had 14/20 for the “risk management and planning” part of the coursework as we had not identified all the possible risks.



3) Risk Analysis Planning

Risk	Likelihood	Impact	Strategy to minimize disruption
1. Not enough time being allocated to certain tasks.	Medium	Moderate	With good planning and maximising use of spare time in-between performing tasks we can fill in free time. In addition, we worked in groups of two on each task so that certain tasks would be completed more quickly.
2. Not knowing enough relevant information about specific areas	Medium	Small	As we have researched this task as a group, we have able to have basic grasp on everything we should know for the task. If there are things that we are not confident in our knowledge of, we will be able to research it later.
3. Key members ill or unavailable at critical times	High	Large	By spreading out tasks equally between two people to one task, this should mean that there will always be at least one person working on each task at any one time.
4. Team members using different versions of programming language.	Low	Large	With good communication skill as we will have constant contact with each other, we will be able ensure the use of the same language in the creation of this software.
5. Ensuring the data of students (fictional or non-fictional) is protected.	Low	Large	No sensitive data will enter or leave our system while this software is being performed as we will use fake student credentials while testing. Also, all parts of the system will be password protected.
6. The university's network and file system fails or is unavailable during maintenance.	Medium	Large	To prevent files being unavailable to us we will have several copies and previous versions stored securely off site in several locations.

Table 2 Risk analysis for the TMS project

Looking back, we had obviously underestimated the likelihood and impact of the risk associated with Risk 1 “Not enough time being allocated to certain tasks” and Risk 2 “Not knowing enough relevant information about specific areas.” Also, the strategies to minimise disruption for these risks had proved inadequate. These two risks were not mutually exclusive since Risk 2 was made worse through Risk 1 becoming reality. Therefore, had we allocated more time to the coding and integration, then the we would have had more time to perform the necessary research to overcome our inexperience with coding and achieve the requirements. We were probably also complacent and, although anticipating these problems, we had not acted quickly enough to minimise their impact.

This relates to a famous quote, “If you don’t actively attack the risks, they will actively attack you” (Gilb, 1988).

3.3 Version Control

When more than one designer is working on a system it is vital that the source code is properly managed to ensure everyone is working on the most recent version. There are increasing numbers of version control software systems available to ensure this happens. The system recommended in my department is GITHub, which is a widely-used web-based service. My team had its own GIT repository which we used for file sharing and version control purposes. I made the repository using my account and by using this we shared all our individual files, code and any documentation. Using GITHub was very effective and efficient because not only could we back up our work, but we could also work outside the university intranet and share files via the internet. We needed to stay in touch and discuss what problems we were encountering and which files were affected, thereby making it easier to make changes and ensure everything in the system worked. The most useful application of GIT was that it overwrote files that had been slightly changed, and showed us the exact changes that had been made, including the newly updated information within the file, and what had been deleted. We also communicated through our chat, and informed one another when we uploaded a new or revised file.

3.4 Software development Techniques

Of the techniques which I was introduced to in lectures the one I found most beneficial was use case models/diagrams and the one I found least useful on this occasion for producing quality software was the class diagram.

A use case is sort of a contract between developers and customers. Therefore, a use cases diagram at a glance puts the requirements into a visual format identifying the interactions needed between an actor and the system. It has been said that “A picture is worth a thousand words”, which is very true for software design. Essentially the pages of text provided for this coursework, outlining the requirements for the TMS, was simply presented diagrammatically on one page. I found the use of the case diagram an easy to understand method of presenting my understanding about what the system needed to do. It also helped me plan the coding and integration of the TMS prototype. I would suggest also that these diagrams would be valuable to non-technical customers and so, help validate their requirements.

Conversely, I found the class diagram was of least help to me when I was coding. I know this diagram should clearly illustrate *relationships* between classes and interfaces and provide detailed information to guide coding and help execute code. However, I found the class diagram my team had produced difficult to implement when coding. From the feedback on our work we realised that when we created the class diagram, it was unclear, for instance we had added a test case which we had not previously defined. Our class diagram was therefore flawed, did not make sense in justifying our design and thus did not help with the system's technical content.

4. My role in the team

4.1 Evaluation of my contribution to the team project – Team organisation

I was the person who presented new ideas and approaches and I also became the specialist debugger of code, helping to provide advice about the code written by others within my team. I was also the person who presented new ideas. As the project proceeded and our skill set and confidence expanded, so did our roles. Thus, whilst I started to provide specialist skills in debugging code, another member assumed more of a role of chairperson, helping to coordinate the work. We therefore all complemented one another.

One problem within our team was that we had underestimated the time needed to complete the tasks, and so time management was an area where our biggest weakness lay. Through encouragement using group chat, I tried to keep spirits up and work hard to overcome our technical inexperience to try to meet our deadlines. When we got together in university I would try to help debug the code written by others. One of my personal weaknesses is communication, and sometimes I would tend to get caught up with my coding and fail to keep the others in the team informed of progress. This is an area I need to focus on in future.

We used modularity to development the system, where we broke the coding task up into smaller more manageable modules. Although we had not appointed a project manager, I offered to do the first part of the coding and suggested how the rest of the work should be allocation to each individual. After each phase of designing, testing and debugging code independently, implementation required integration of the complete code into the system using GIT for version control. Ultimately, I was given responsibility for integrating the complete TMS prototype in GIT after the rest of the team had written their code and run their test cases. Unfortunately, towards the end of the project we came across bugs which slowed us down and meant the testing could not be completed as required.

4.2 Evaluation of my contribution to the team project – technical organisation

For Coursework 1, it was agreed that I be assigned the roles as shown in Table 3. The table also summarises my technical contributions to the tasks.

ID of coursework	Description of task	My contribution
Coursework 1 Part A: Requirements specification, Project Plan, Risk Analysis and Draft Ideas	Functional requirements (FRs) with brief descriptions (MoSCoW notation and write so they can be validated / acceptance tested) Non-Functional Requirements (NFRs) with acceptance criteria (write so they are testable)	Myself and another member of the team worked as a pair using MoSCoW (Must haves, Should haves, Could haves, Won't haves) to prioritise the FRs of the system that we needed or could possibly implement, in order to make the system perform properly. Ultimately, we wanted to design a quality application to meet all requirement specified by COMSCI in the brief. Unfortunately, our feedback indicated that we had failed to write them with acceptance criteria. Myself and a team mate also identified all the NFRs. These are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours (which relate to FRs). We used examples from other software to give us a good idea of what to include. But we failed to write these with quantifiable metrics against which to assess the NFRs.
Coursework 1 Part B Task 1: Develop a Use Case model to capture the major functional requirements of the ` Personal Tutor Management system	Include in the use case diagram any possible relationships between the use cases (namely, include, extend and generalise). For each use case identified, write a brief description of the use case and create a step-by-step outline of the basic flow of events. (It is expected that every member of the group will write ONE use case each).	I had the job of identifying and defining the relationships between the use cases. I used my lecture notes and information from the internet to help me devise the most sensible relationships between the use cases. I researched methodology for use case models and wrote briefly about the flow of events which related to the use case called “assign student to tutor”, clearly specifying the step-by-step process which occurs during the uploading of the CSV file and assigning the students to tutors automatically based on their course.
Coursework 1 Part B Task 2: Develop a UML Class diagram for your system	For each class include representative attributes and methods. Describe how your class diagram will as support ALL the major system functional requirements, captured in the use cases you identified above. This description should be a maximum of two pages.	As a team, we struggled to create the UML class diagram. It was hard for us to link the processes and use cases by relationships. We had difficulty justifying our design through a UML class diagram and failed to appropriately use the different types of class relationships. Because of this, our diagram failed to prove with any certainty whether our design supported all the major system functional requirements.

Table 3: Showing my contributions to Coursework 1

Coursework 2: develop a working prototype of the TMS using Python GUI Programming (Tkinter), covering all major functional requirements with and a set of test cases to ensure validity.

I was responsible for writing and testing the code for uploading an excel or CSV file containing information about new students and assigning these new students to personal tutees. Firstly, therefore I needed to think of a way to upload an existing CSV file to my system. I had to make sure it would recognize and accept the file, read the information and extract it in a suitable format to be used for other processes. This was a vital part of the system as all the other aspects of the system require the information from the CSV to be uploaded correctly if the whole system was to function as required. Firstly, I tried to design the system so that the user would be asked to type in the name of the file into a submit box in order find it on the system and access it. This was not very user friendly, so in order to do this more effectively, I needed the user to be able to search through directories to find the directory with the CSV file in. So, I then decided to code the system so that the file could be selected by browsing files on the computer via file explorer. This made choosing the correct file a lot more efficient and made the program much simpler to use. I then coded the next bit of the program which was to auto assign students to personal tutors based on their course. I did this by simply comparing the course of the student CSV to the course of the tutors CSV, and then putting together a dictionary with the tutors and their new tutees. When I tested the system, I managed to get all of the steps in the basic flow to pass, and documented this to demontrate validation. I therefore had "excellent" for validation during the testing for this part of the prototype. However, my "Clarity of Instructions" and "Test Case Relevance: (All essential steps – basic flow + interface images)" was marked as adequate. Unfortunately, this aspect of the work suffered due to lack of time.

5. Quality Criteria

Here I will use the hierarchical quality model by McCall (1977) to highlight where my part of the TMS showed consideration of three quality criteria characteristics: completeness, consistency and reliability. McCall's model is appropriate here as it is aimed at system developers and is designed for use during the development process.

I produced python code that allowed me to import CSV data and created a section of code to search for specific information within the file, whilst also using the python module tkinter to produce a simply GUI interface for the user to work the system. In order to maximise correctness, I needed to ensure optimum completeness and consistency. Correctness is the extent to which a program fulfils its specification. One aspect of this is completeness, which McCall (1977) refers to as "Those attributes of the software that provide full implementation of the functions required". In other words, completeness is the degree to which a full implementation of the required functionality has been achieved using validation testing to check the program meets all functional requirements. It can be measured through a metric relating to a checklist of nine items. The metric can be calculated as the sum of the scores of the following applicable elements, divided by the number of applicable elements 1) unambiguous references (input, function, output), 2) all data references defined, computed, or obtained from an external source, 3) All defined functions used, 4) All referenced functions defined, 5) All conditions and processing defined for each decision point, 6) All defined and referenced calling sequence parameters agree, 7) All problem reports resolved, 8) Design agrees with requirements and 9) Code agrees with design.

My code met all of these items. From the beginning I had studied the required functionalities and made sure that I designed my code so that all functionalities were fully implemented. For example, to ensure my code covered items 8 and 9, I needed to show that CSV files could be imported into my system, thus I created test CSV files with 8-10 rows of data containing the relevant student and tutor information and tested the specific import functionalities of the system. As per requirements, I also ensured that my code uploaded the data into a dictionary in a format appropriate for manipulation or editing. I also performed validation checks to ensure the code I had written efficiently and effectively assigned new students to personal tutees. I tried to perform "just enough" testing in the plan without over or under testing.

Consistency can be quantified in terms of a metric based on items relating to Procedure Consistency Measures and Data Consistency Measures, which relate to standardising design and implementation at system level (McCall, 1977). To ensure consistency, I tried to use a uniform design and notations within my code even though there were multiple implementation techniques needed to perform specific functions. I thus used sensible variable names and comments in order to explain how certain bits of the code work. |For example, I created two dictionaries with variable names 'tutorAssignedStudents' and 'tutor_list'. These dictionaries were named sensibly as the tutorAssignedStudents' dictionary was used to hold information of students that had been assigned to tutors automatically based on their course and the 'tutor_list' was used to hold the information of the tutors from the CSV file. I created these dictionaries at the beginning of the python file and used the global function to integrate them into other functions. I also frequently used 'def' functions, which made it easier to consistently call functions multiple times and make my code a lot more efficient. My main responsibility was to design a function to auto assign students to tutors as the CSV file was uploaded. I did this by looping through the tutor_list and student_list and using index numbers to compare the course ID of the students in the student CSV file to the tutors in the tutor CSV file. I then appended the students which could be auto assigned to the correct tutor matching their course using the ‘append’ function to append them to the ‘tutorAssignedStudents’ dictionary. I also, used integrated loops in my function in order to make the code more efficient and reliable. The following code illustrates all of these points.

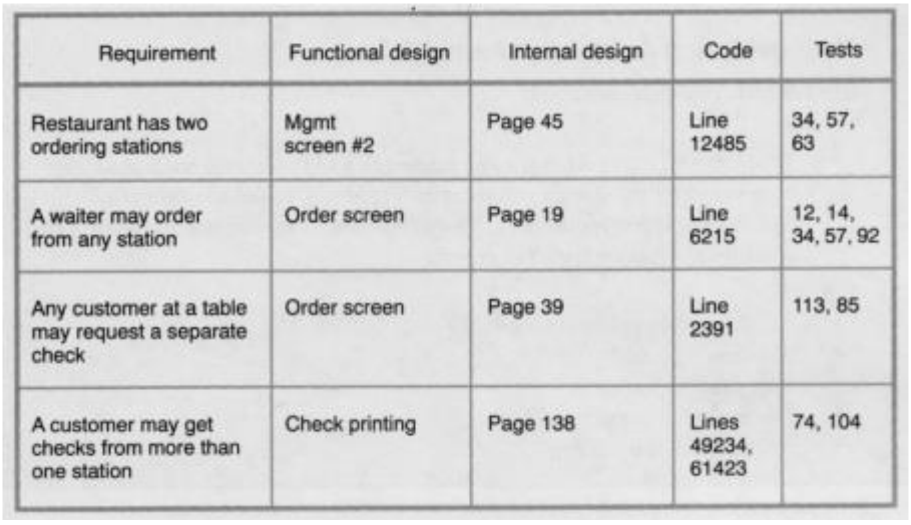
```
global tutorAssignedStudents
    lastAssignedStudent = 0
    recordsAssigned = []

    for tut in tutor_list:
      assignedStudentRecordCount = 0
      assignedStudentCount = 0

      for stu in self.student_list:
        assignedStudentRecordCount += 1

        if (len(recordsAssigned) > 0) and not (assignedStudentRecordCount in recordsAssigned):

          if tut[2] == stu[5] and stu[6] == tut[3]:
            assignedStudentCount += 1

            recordsAssigned.append(assignedStudentRecordCount)

            tutorAssignedStudents.append([stu[0], stu[1], stu[2], stu[3], tut[1] + " " + tut[0],
stu[5], stu[6], stu[7]])
            lastAssignedStudent = assignedStudentRecordCount

            if (assignedStudentCount >= 6 and tut[5] == 'PT') or (
                    assignedStudentCount >= 11 and tut[5] == 'FT'):
              break
        elif len(recordsAssigned) == 0:
          if tut[2] == stu[5] and stu[6] == tut[3]:
            assignedStudentCount += 1

            recordsAssigned.append(assignedStudentRecordCount)

            tutorAssignedStudents.append([stu[0], stu[1], stu[2], stu[3], tut[1] + " " + tut[0],
stu[5], stu[6], stu[7]])
            lastAssignedStudent = assignedStudentRecordCount

        if assignedStudentCount == 120:

          break
```

Additionally, for notations and variable names to remain the same across the team, good communication between members was needed to ensure consistency for the whole system.

Reliability refers to the ability of the system not to fail. It covers accuracy, robustness, precision and tolerance. Another important area of reliability is simplicity, which refers to the ease with which the software can be understood. It requires a software developer to ensure that those attributes of the software that provide implementation of functions do so in the most understandable manner. That means avoiding practices which increase complexity. To maximise simplicity during the writing of my code, I tried to follow a good programming style, I also kept the code simple and well structured. Examples of this were described previously, for example to make my code efficient yet very simple, I used different coding techniques including applying multiple functions to make my code more structured and used global variables to avoid creating too many variables with unsuitable variable names. I also used inbuilt modules in python such as using the import function in order to make coding certain requirements easier, such as when I needed to code for importing the CSV file.

6. Recommendations for undertaking similar team projects in the future

My first recommendation would be regarding the software development process model. I would suggest that further focus be given to quality criteria. In particular, in order to easily identify whether there are any missing functionalities or requirements, a traceability matrix could be put together. This would give clear documented evidence linking the software components to the requirements. Documenting cross reference links between requirements and the code could be achieved by setting up a table in MS Excel where system requirements would be in rows and links to the code line and validations tests covering that requirement would be in columns. An example of this is given below.



Requirement	Functional design	Internal design	Code	Tests
Restaurant has two ordering stations	Mgmt screen #2	Page 45	Line 12485	34, 57, 63
A waiter may order from any station	Order screen	Page 19	Line 6215	12, 14, 34, 57, 92
Any customer at a table may request a separate check	Order screen	Page 39	Line 2391	113, 85
A customer may get checks from more than one station	Check printing	Page 138	Lines 49234, 61423	74, 104

Table 3 An example of a software requirement tracing matrix (from HubTechInsider (2011))

Using this matrix has several benefits including a) To make sure that all requirements are included in the test cases b) To make sure that no unnecessary features or extra functionality have been developed thus avoiding wasting time and effort and c) If there is a need to change a particular requirement then it is easy to determine which test case needs to be updated.

My second recommendation would be to improve team working by being more aware of theories and applying them from the outset. Through writing this reflective report I became aware of the benefits of having prior knowledge about theories such as Tuckman's team organisation theory and Belbin's team theory. It is important to establish from the start the natural preference that each person has with regard to the various team roles required to complete a project. Then the roles can be assigned so that there is a balanced team. Also, being aware of the Forming, Storming, Norming and Performing stages of Tuckman's theory would help the team members better understand the dynamics of the team as a whole, and when the team is stuck in the storming stage the project manager could drive the project forward by motivating and empowering the other members of the team. Where necessary this might mean redeploying resources to meet the project requirements in a timely manner, such as reorganising roles so that relevant expertise is shifted to help overcome problems and meet deadlines.

My third recommendation is to improve project management through better, more realistic planning. This would be reflected in the Gantt chart. In creating the chart, it would be vital that all members of the project team contribute and ensure that they do not underestimate the amount of time it takes for them to complete each deliverable of the project. The chart must also accurately show links and relationships between the scheduled tasks. The quality of the data inputted is vital to avoid the "Rubbish in, rubbish out" syndrome. Careful planning would also need to link with risk management so that all steps can be taken to mitigate against potential problems. Keeping to the deadlines is vital for timely delivery of the final working system. The project manager should take overall responsibility for monitoring project completion against the Gantt chart and amending the timings as necessary.

Finally, in terms of my role, if faced with another similar project, I would suggest a project manager be appointed and I would put myself forward for the role. As I am inexperienced in leading a team, I would need to use appropriate leadership skills to ensure we met the needs of the specification in a timely manner. I would therefore extend my knowledge of leadership styles, conflict management, team motivation techniques and negotiation skills. I would also need to learn how to optimise communication, cooperation and collaboration across the whole team. My natural leadership style would be “leadership by example” and “visionary leadership”, but I would need to learn other styles so that I could tailor my approach depending on the project and team members as described in the publication by Müller and Turner (2007). For example, a more autocratic, goal-oriented leadership style would be appropriate as deadline’s approached.

7. References

Belbin, R. M. (1981) *Management Teams - Why they succeed or fail*. Butterworth Heinemann

CM1202 lecture notes (2017) *Developing Quality Software*: Learning central [online] https://learningcentral.cf.ac.uk/

Gilb, T. (1988). *Principles of Software Engineering Management*. Addison-Wesley.

HubTechInsider (2011) *What is software traceability? What is a software requirements traceability matrix*? July 12, 2011 [online] https://hubtechinsider.wordpress.com/2011/07/12/what-is-software-traceability-what-is-a-software-requirements-traceability-matrix/

Largent, D and; Lüer, C (2010) *You mean we have to work together!?!: a study of the formation and interaction of programming teams in a college course setting*. Proceedings of the Sixth international workshop on computing education research, 09 August 2010, pp.41-50

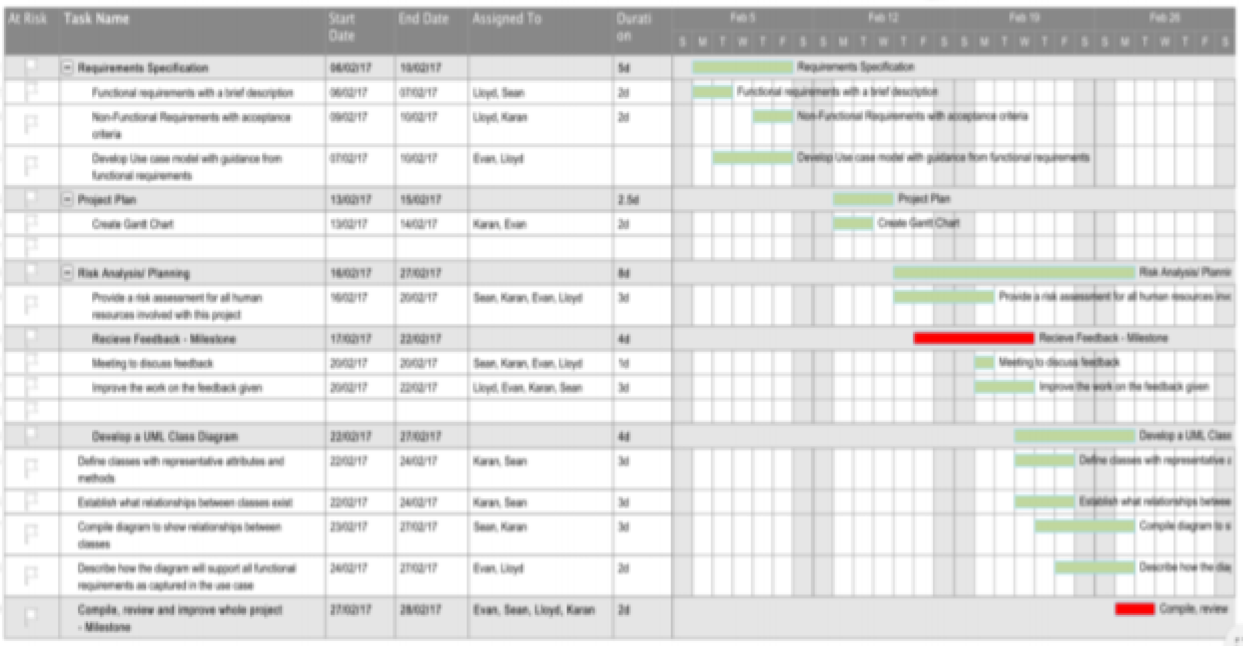
McCall, J. A., Richards, P. K., and Walters, G. F., (1977) Factors in Software Quality, *Nat'l Tech. Information Service*, no. Vol. 1, 2 and 3.

Müller, R., and Turner, J. R. (2007). Matching the project manager's leadership style to project type. *International Journal of Project Management*, 25(1), 21–32.

Tuckman, B. W. (1965) Developmental sequence in small groups. Psychological Bulletin, 63, 6, pp 384-399.

van Vliet, Hans (2008) *Software engineering : principles and practice* Chichester ; Hoboken, NJ : John Wiley & Sons 3rd ed. c2008

8. Appendix – Gantt Chart for Team 23



	At Risk	Task Name	Start Date	End Date	Assigned To	Duration
1		Requirements Specification	06/02/17	10/02/17		5d
2		Functional requirements with a brief description	06/02/17	07/02/17	Lloyd, Sean	2d
3		Non-Functional Requirements with acceptance criteria	09/02/17	10/02/17	Lloyd, Karan	2d
4		Develop Use case model with guidance from functional requirements	07/02/17	10/02/17	Evan, Lloyd	
5		Project Plan	13/02/17	15/02/17		2.5d
6		Create Gantt Chart	13/02/17	14/02/17	Karan, Evan	2d
7						
8		Risk Analysis/ Planning	16/02/17	27/02/17		8d
9		Provide a risk assessment for all human resources involved with this project	16/02/17	20/02/17	Sean, Karan, Evan, Lloyd	3d
10		Recieve Feedback - Milestone	17/02/17	22/02/17		4d
11		Meeting to discuss feedback	20/02/17	20/02/17	Sean, Karan, Evan, Lloyd	1d
12		Improve the work on the feedback given	20/02/17	22/02/17	Lloyd, Evan, Karan, Sean	3d
13						
14		Develop a UML Class Diagram	22/02/17	27/02/17		4d
15		Define classes with representative attributes and methods	22/02/17	24/02/17	Karan, Sean	3d
16		Establish what relationships between classes exist	22/02/17	24/02/17	Karan, Sean	3d
17		Compile diagram to show relationships between classes	23/02/17	27/02/17	Sean, Karan	3d
18		Describe how the diagram will support all functional requirements as captured in the use case	24/02/17	27/02/17	Evan, Lloyd	2d
19		Compile, review and improve whole project - Milestone	27/02/17	28/02/17	Evan, Sean, Lloyd, Karan	2d

*any cropped words are present under the table heading: 'Task Name'