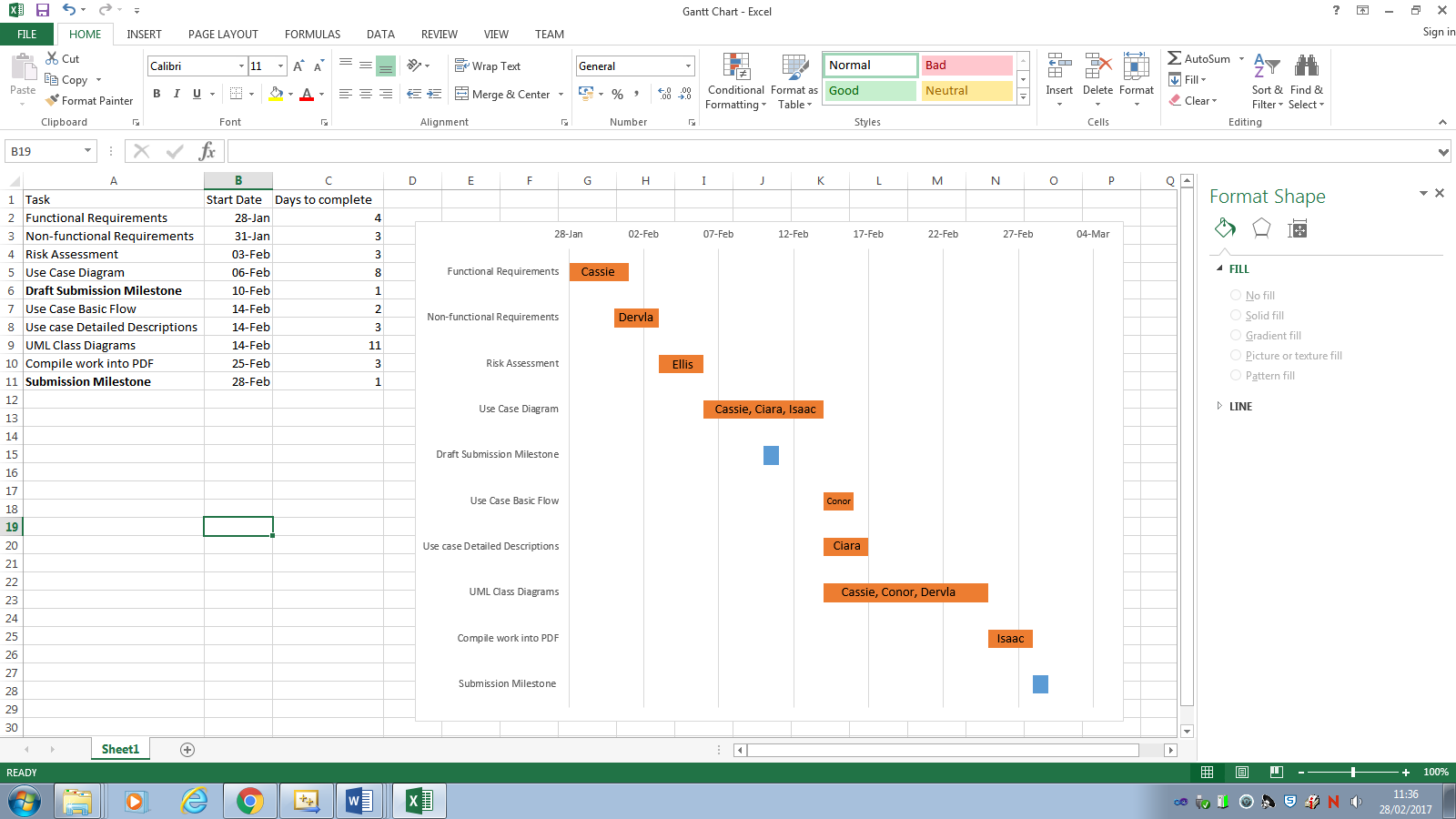
Group 16

Gantt Chart Planning

The first thing we needed to do in order to structure the project and manage our time efficiently was create a Gannt Chart using Excel. In doing so we would be able to estimate the time required in order for us to complete each of these tasks. The tasks set before us included listing functional and non-functional requirements, risk assessment, creating a use case diagram, making and UML Class diagram, compiling work from the planning stages into a single PDF document, developing our program and debugging the program. Also included in the Gantt Chart were milestones. In this case, these were our submission deadlines.



# Functional Requirements

The second task set before us was to produce a list of all the functional requirements for the program that we were designing. These functional requirements were critical to the functionality of our program, as they would determine exactly what it would do, and how it would go about doing it.

We split our list into requirements that must be met, should be met, could be met and would be met.

## Must:

* Read a list of staff members that are able to have tutees from a .csv file.
* Store a list of those staff members and their details which can be easily viewed.
* Allow staff members’ details to be edited on the system.
* Allow staff members’ details to be deleted from the system.
* Read lists of computer science students into the system from a .csv file.
* Store a list of those students and their details which can be easily viewed.
* Allow students’ details to be edited on the system.
* Allow students’ details to be deleted from the system.
* Allocate students to tutors in small groups. These groups should be allocated based on the speciality of the tutor’s research specialty.
* Store a list of those groups which can be easily viewed.
* Read a list of courses into the system from a .csv file.
* Store a list of those courses and basic details which can be easily viewed.
* Allow groups to be edited, and for students to be deleted or added to those groups.
* Be able to query the system to find a specific student.
* Be able to query the system to find a specific staff member.
* Be able to query the system to find a specific group.
* Be able to query the system to find a specific staff member’s group
* Be quicker than the current manual system to use and maintain.
* Pass all acceptance testing.

## Should:

* Keep track of the number of students in each year of university; first year, second year, third year, placement year, fourth year and postgraduate.
* Allow students to view their tutor groups.
* Create a size-limit on groups, giving each group a maximum number of students.
* Group size should depend on whether the tutor is part/full time.
* Have different levels of access for individual users so private information can only be accessed by the admin.

## Could:

* Used efficient, advanced algorithms for assigning students
* Allow students to request reassignment

## Would:

* Create a login system for students to view their tutor groups.
* Allow students to be able to contact other members of the group when they login.
* Have a help button so when the user clicks on it, they are shown what each button does. This will help them manoeuvre around the system more freely.

Non-functional Requirements

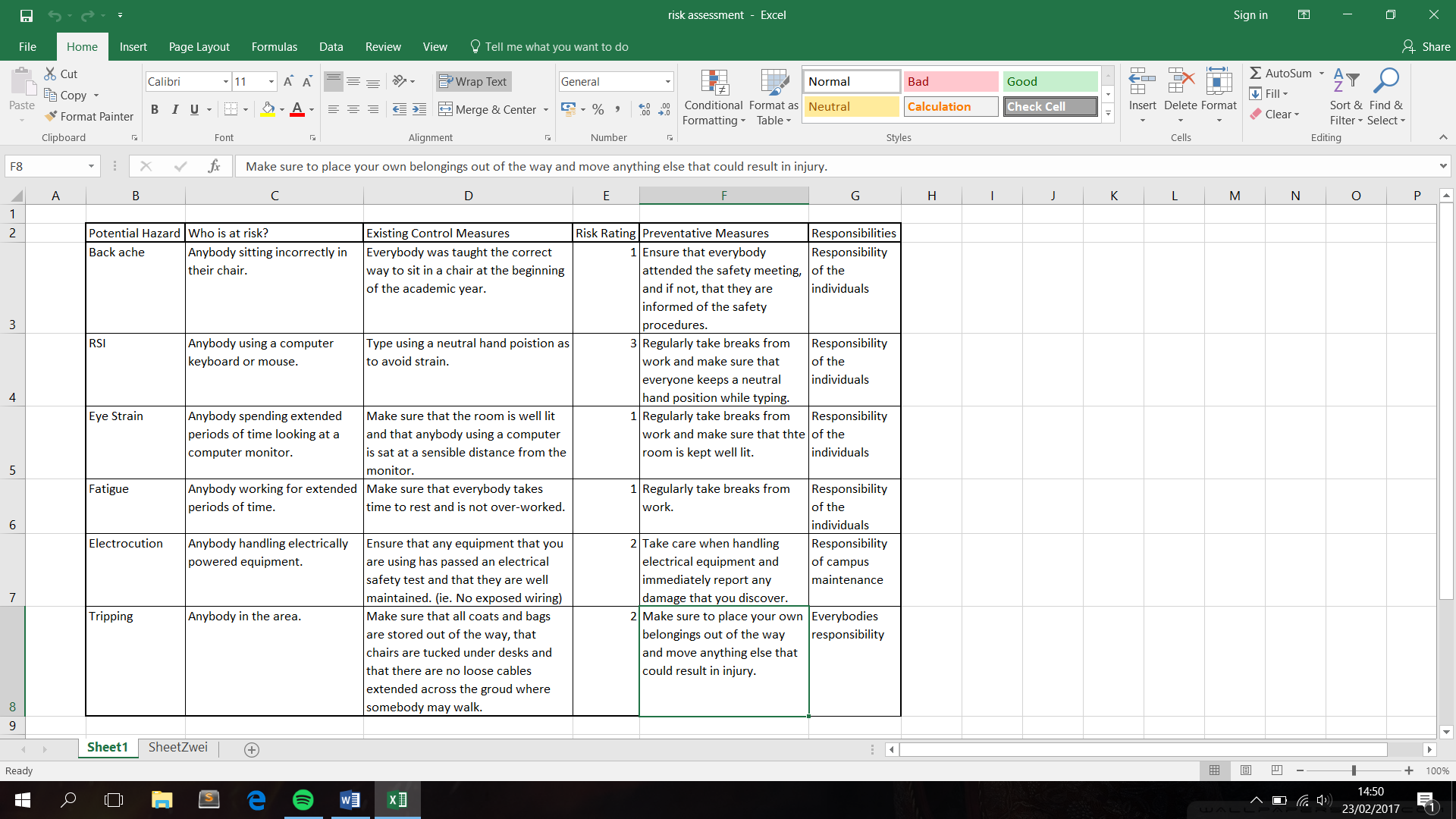
Part of the Project involved making a list of the non-functional requirements of the system. When developing these requirements we took into account the product requirements in terms of performance, reliability and usability, the software and hardware required for the system and the security, privacy and interoperability of the system.

What follows is the final list of non-functional requirements that were agreed upon.

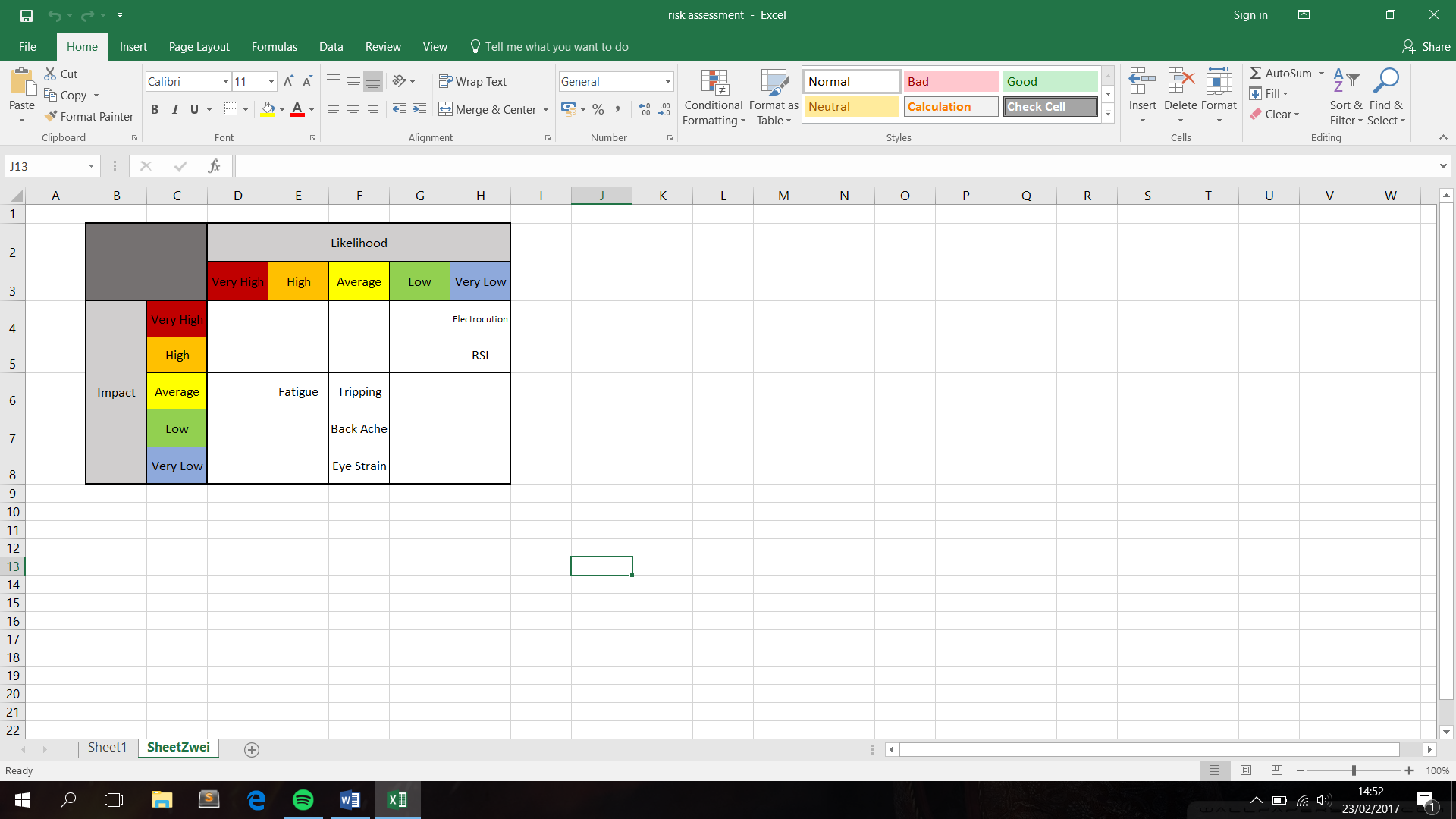
* The system must not take up too much memory so should not exceed 200KB of memory.
* The program must not violate any current firewalls or safety procedures currently in place on the university system.
* The system should function correctly on all operating systems installed in the university.
* The system should be quicker than the previous manual system. This requirement will be fulfilled if all the staff, student and course details can be added to the system within forty-five minutes.
* The staff and students should be allocated to tutor groups accurately with no incorrect allocations eg. Allocated twice or never.
* The administrator should be able to use all system functions intuitively with little to no training.
* The system should be able to handle up to 900 students and staff in the database.
* The database should be able to handle up to five users at the same time.
* The database should update immediately when edits are made to information.
* The system should be compatible with Excel spreadsheets and read information imported from there into the database.
* The system should not contain any sensitive personal information.
* The system should not allow students to view other students’ details.
* The system should not hold the administrators’ passwords in plaintext.
* The system should allow an administrator to log in.
* The system should allow an administrator to log out.
* Only the administrator should be able to edit the system.
* The system should have no serious bugs or errors which hinder performance.
* The system should respond to queries of the system within 90 seconds.
* The system should have a visually pleasing design.
* The system should use high contrast colours to reduce eye strain.

Risk Assessment

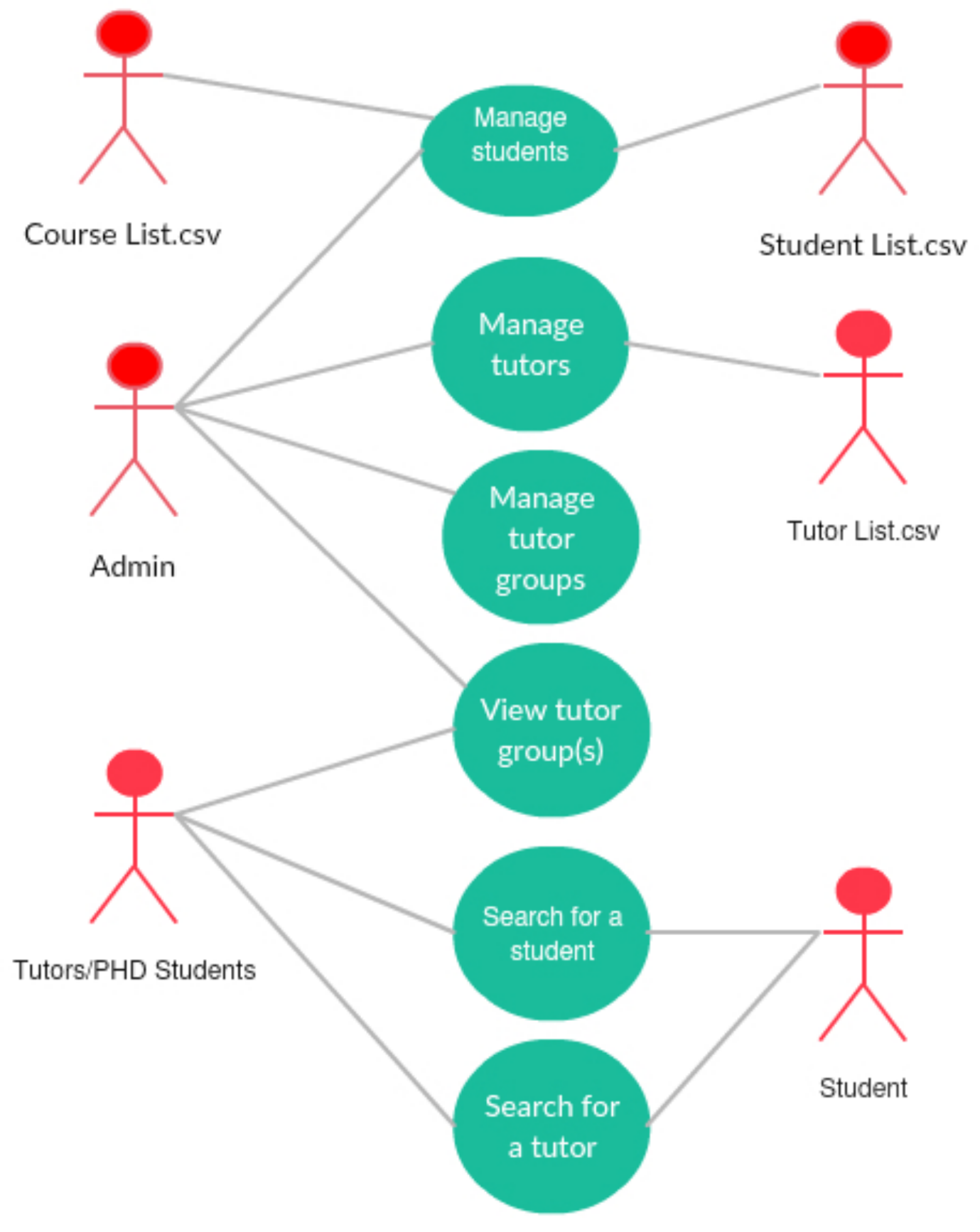
One of the stages of planning was to conduct as risk assessment to identify all the possible dangers involved with our project, and determine ways in which these potential dangers could be prevented or minimised. Although the risks involved with our project, and the field of computer science in general are far and few in between, there are a few risks that can present a significant threat. For instance, RSI or chronic back pains. Threats such as these can be easily avoided if people sit correctly and comfortably, and ensure that they type with neutral hand positions. Although much more extreme and unlikely to occur in university labs, things like electrocution also had to be taken into account while conducting the assessment.



In order to rank all of the identified risks more clearly, we created a colour coded table that showed all of the risks and ranked them by severity and likelihood. In doing so, we were able to easily recognise which risks required us to take action. These were the risks that appeared closer to the top left of the table. Those nearer to the bottom right of the table were insignificant or trivial by comparison.



Use Case Diagram



Use Case Descriptions

***Manage Students***: Within this use case we’re able to add, delete, search and edit students. When the students are added the system reads in a list of students and their details from a ‘StudentList.csv’ file and inserts them into the database. It also reads from a ‘CourseList.csv’ file. When a student is to be searched for, they will be located according to a specific key i.e Student Number. When the students are deleted and/or edited from database they must be searched for first. If it is decided that the student will be deleted the specific student will be removed from the database. If the student is to be edited from the database the student details will appear and they will be able to be written to, however this task must only be performed by the administrator.

***Manage Tutors:*** Within this use case were able to add, delete, search and edit students. When a new tutor is added into the system, it reads a list of tutors and their details from a .csv file and inserts them into the database. When the tutor is to be searched for, they will be located according to a specific key i.e Tutor Number or their name. When the tutors are deleted and/or edited from the database they must be searched for first. If It is decided that the tutor will be deleted the specific tutor will be removed from the database. If the tutor is to be edited from the database the tutor details will appear and they will be able to be written to, however this task must only be performed by the administrator

***Manage Tutor Groups:*** This use case is only managed by the administrator. They admin is able to add and remove students from a tutor group, search for a student and see which tutor group they belong to. They can also perform the opposite of this; search for a tutor and see which students are in that group. The administrator is also able to reassign students and tutors to tutor groups through the use case. Students may request to be reassigned to a new tutor or they may be reassigned to a new tutor because they have changed course i.e computer science to computer science with security and forensics. Therefore the student would be reassigned to a new tutor who would have research specifically for this field.   
Additionally the tutor may feel the need to request for a student to be reassigned to another tutor, this may be because they feel they have taken on too many students for their schedule or because they feel a student may be better benefited with another tutor if they have shown keen interest in a specific field or area that another tutor specialises in.  
This use case must only be accessed by the admin because the fields and .csv files involved contain private information about both tutors and students.

***View Tutor Groups:*** This use case coming from the perspective of the ‘Student’ actor they are able to search for their tutor group and see all other tutees within it. This may be because the student needs to contact another member of their group so displays their email for communication purposes and for the purpose of privacy the student receiving the email must firstly confirm that they want to accept a message from this person. The student is also able to view their tutor group so that they are able to see the name of their tutor and contact details for their tutor including the office room of their tutor in case the need arises that the student needs to see their tutor or for a meeting.

This use case coming from the perspective of the ‘Tutor’ actor, means that the tutor is able to search for their tutor group to see all members of the group or to see what year of university the group is in. The tutor is also able to contact all members of their group collectively for example if they want to arrange a meeting. The tutor is also able to contact each group member individuality if they need arises that they need to discuss something specific with them.

***Search for a Tutor:*** From the perspective of the ‘Student’ actor, they are able to search for their own tutor. This is for the purpose of displaying the contact information of their tutor in case the need arises that the student needs extra help or advice. It also allows the student to see the room number of the office that the tutor has if they have a meeting and it is held there. The tutor information held within the university includes the qualifications that the tutor holds and this is useful for the student to be able to view this of the tutor because they can ask the tutor for advice on what path to take or for general advice about their course.

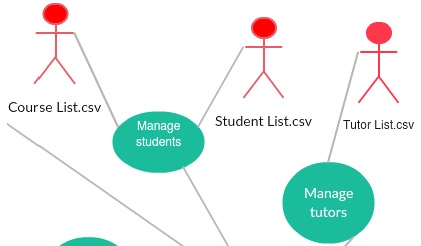
From the perspective of the ‘Tutor’ actor, they may need to search for another tutor because they feel like they need advice on how to help their own tutees or they’re interested on how other tutors interact with their students. This may benefit the tutor as they might get ideas and form a better relationship with their tutees. Tutors may need to search for another tutor because they wish to contact them about how they manage both their research and tutor group.

***Search for a Student***: This use case is for the ‘Tutor’ actor. This enables them to search for a student, through doing this they are able to see their student number, they forename and surname, what year they’re in, what course they are on, what tutor they have and their email address. This may be beneficial because if a student comes to them unsure of who their tutor is they are able to search for it. It is also a necessary use case for the tutor because they are able to search for the email of a student.

Detailed Use Case Descriptions

Basic Flow of the Use Case Diagram

­­­The flow of the data begins with two files, ‘StudentList.csv’ & ‘CourseList.csv’. These lists contain all of the necessary information about each of the courses and each individual student. For the student list, it will include details such as their unique student ID along with what course they are currently attending and which year group they are in. The course list file will contain information about each course, including details about the course such as academic staff involved with the subject and the number of students within that course. Both of these files get processed into the ‘Manage students’ use case, which processes the information such as adding, deleting or expanding upon it.

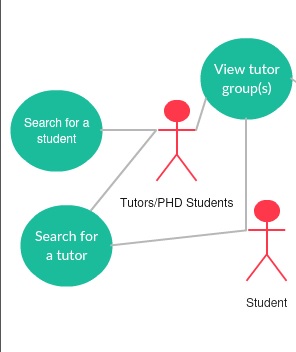


Alongside the previous two lists is the ‘TutorList.csv’ which is a file containing all the necessary information about each tutor, such as whether they are part-time, full-time, PhD students and so on. This list gets processed through the ‘Manage tutors’ use case, which allows the admin to add, delete or modify the contents of the tutor list. This then links to the same actor as the ‘Manage students’ use case.

The previous two use-cases move forward to the actor ‘Admin’. The admin actor is responsible for managing all of the tutors and students as well as assigning them to their own individual groups. This is shown by the flow of data from the admin actor to the use-case ‘Manage tutor group’.



This shows the admin has full access to modify and manage the tutor groups which includes one tutor and several students (tutees). The other direction in which data flows from the admin is to the ‘View tutor groups’ use-case, showing that the admin modifies the tutor groups and then these tutor groups are viewed by the tutors and students alike.



From here, the student actor and tutor actor can both view the tutor groups, which can only be modified by the admin. Both the student and tutor actor can gain access to the ‘Search for a tutor’ use-case, as it is a necessary function for both actors. The tutor actor also has exclusive access to the ‘Search for a student’ use-case, which allows them to find any student by name or ID. The same can be done in the ‘Search for a tutor’ use-case which allows any actors linked to search for a tutor by name.

UML Class Diagram