# Part A – Developing Quality Software

# Actors

* **Student**: A student is a person with a valid student ID who is registered at the college. The student IDs are unique.
* **Tutor**: A tutor is a member of staff at the university with responsibility for a group of students.
* **Course List**: A course is a program of study a student can undertake identifiable by a unique ID.
* **Spreadsheet**: This spreadsheet details information of each student. The system must read in from this file the details of all new students so they can be allocated to the correct tutor.
* **Administrator**: The administrator is a person with a unique ID and password who has authorisation to maintain and view the information in the system

# Brief Descriptions

### Maintain Student Information:

This use case allows the administrator to maintain student information in the system. This includes adding, modifying and deleting students from the system.

### Maintain Tutor Information:

This use case allows the administrator to maintain tutor information in the system. This includes adding, modifying and deleting tutors from the system.

### Search for Student:

This use case allows tutors or the administrator to search for a student using the student’s name or student ID. Once a student is found the tutor or administrator can view the student’s details and view the tutor group they belong to.

### Search for Tutor:

This use case allows the administrator to search for a tutor using by name or by research field. Once found the administrator can view the details of that tutor and the details of all the students within their personal tutor groups.

# Functional Requirements

## 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.

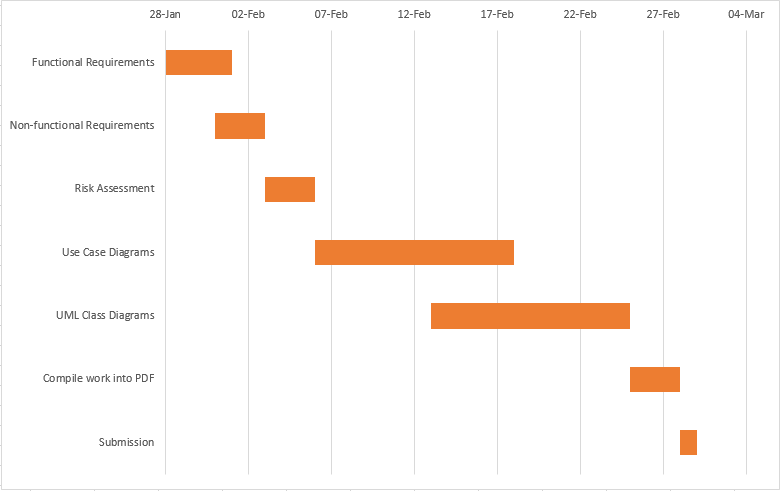
# Non-Functional Requirements

* Be user-friendly.
* Have a clear and concise colour scheme.
* Be easy to keep up to date.

**Gantt Project**

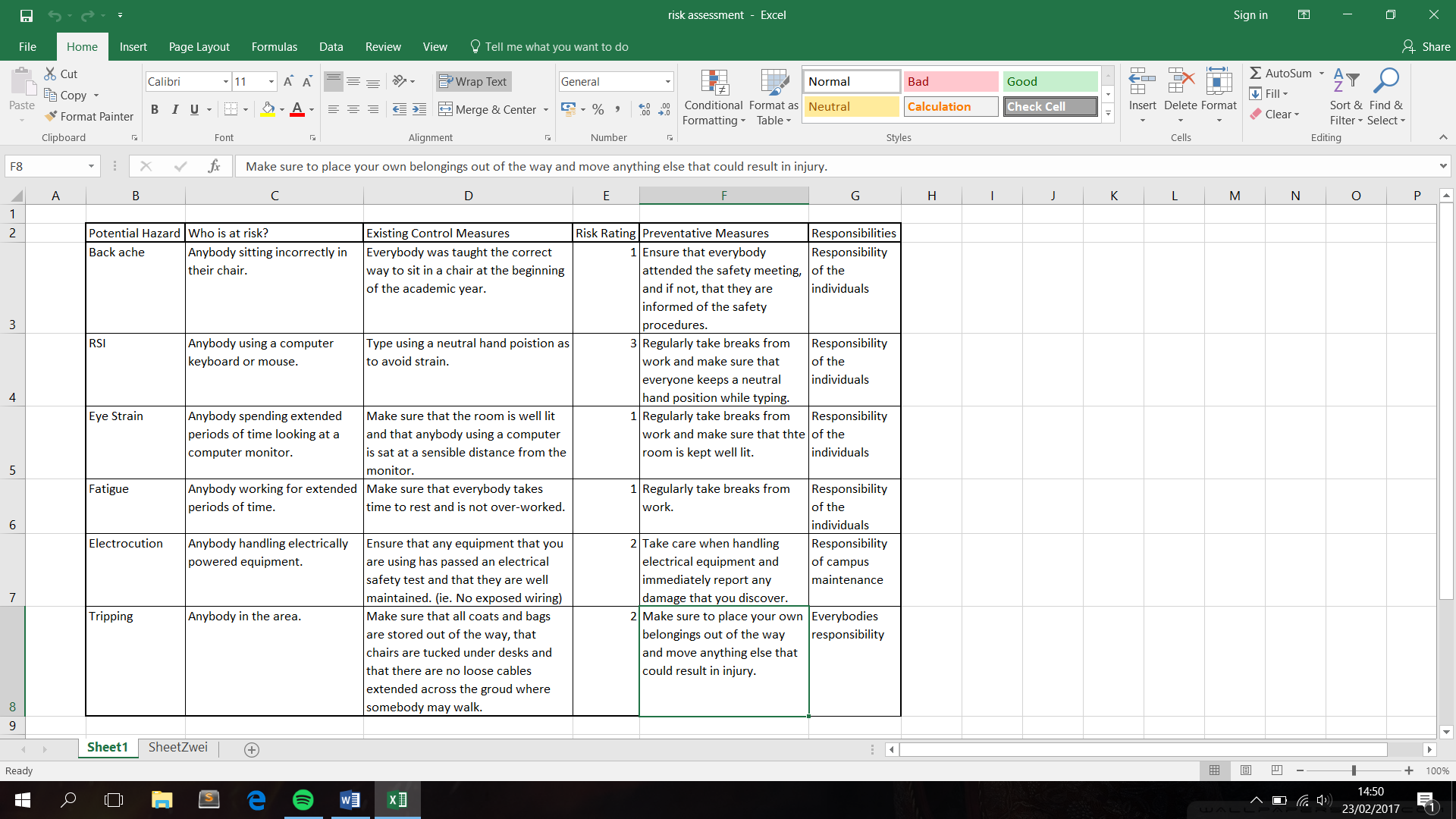
Gantt Chart Planning

The first thing we needed to do in order to structure the project and manage our time efficiently was create a Gantt 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. **INCLUDED STUFF FOR THE ACTUAL PROGRAM DEVELOPMENT HERE BUT HAVEN’T ADDED IT TO THE GANNT CHART. NOT SURE IF I SHOULD OR NOT YET.** Also included in the Gantt Chart were milestones. In this case, these were our submission deadlines.



**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.

