Team Project FAQs

# Gitlab/github

You will get a university account.

# Team Project General feedback 2021 on Moodle

If you have not already done so, you must all read the Team Project General feedback 2021. You can find it on the Moodle page. It will be very helpful.

|  |
| --- |
| Planning |
| Planning is very important. **Can you just go and build a house without prior preparation and planning?** No. The same here. |
|  |
| 1.1 **An analysis of the project assignment brief** (try to understand what is required - what they ask you to do). This part should include but not be restricted to the following (write as you go - my best advice): |
| Interpretation of the project assignment brief (write draft Introduction chapter) |
| Aims of the project assignment (write draft Introduction chapter) |
| **Information gathered to clarify the brief**, e.g. background research, survey, interview, such as looking at similar problems, products (write a draft in the Background chapter) |
| **Functional and non-functional requirements** (you already have the minimum provided, but you could add) (write a draft in the Background chapter after reviews) |
| Identification of resources and materials required and how they will be accessed/obtained. For example, the development environment. (**this will be in the Design chapter**) |
| Identification of information sources to be used (references/bibliography - **remember you will cite them in the report Introduction/Background chapters**) |
| List all the tasks and share them amongst the members. |
| 1.2 **Project plan (Gantt chart) to be included as appendix**— Production of a formal plan to undertake the project with realistic timescales and identifying: |
|  |
| 1.2.1 - Schedules for each stage and overall completion |  |
| 1.2.2 - Milestones and deliverables |  |
| 1.1.4 - Main tasks |  |
| 1.1.5 - Resources |  |
|  |  |
| 1.3 **Solution Plan: Analysis and Design** |  |
|  |
| 1.3.1 - Undertaking the analysis using appropriate techniques (mentioned above, initial background survey, etc.) |  |
| 1.3.2 - Evidence for the analysis and design of the prototype application for the given brief (**wireframes/storyboards, user stories, flow diagrams, system architecture, ER diagrams, etc.**) (**All these diagrams are for the design chapter**). |  |

# Design chapter and what design artefacts you should think about

First, I would suggest you separate the Design and Implementation into different chapters.

The Design Chapter should not be used as a place for dumping diagrams. It includes all the necessary diagrams and their description. The wireframe/prototype should show how the system will look like BEFORE implementation. Because of page limits for the main part of the report, you can put all the diagrams in the appendix; however, make sure to signpost them in their corresponding section of the Design chapter. For example, if you have described the components of the System Architecture diagram but there is no space because of page limits, you can put the system architecture diagram in the Appendix section. However, you will write something like this "The system architecture **(see Appendix 2)** is composed of blah blah blah….". To summarise, each diagram should be explained in its corresponding design section even if the diagram itself is in the appendix section.

So, the design chapter describes the system on a high level and could include

* **Use Case diagram**
* Sequence diagram
* Activity diagram
* **Relational Schema (ER diagrams)**
* Flow chart
* **System Architecture diagram**
* (class diagram)
* Site map
* **Wireframes/storyboards**

In this chapter, you should also mention the technologies used and the software development methodology.

# Team project Report - Implementation Chapter

A mere description of the different features or functions will cost you. It should present the technical implementation of the product, including a section that presents the final output (UI). The implementation section should include more specific libraries used to implement specific features, for example, which library is used to create the visualisation graphs…. The implementation should be specific on the libraries used and for which specific features. For example, if you write, "Spyder integrates with several prominent packages in the scientific stack was used to develop this application", you will lose some marks because I do not know which of these packages you used and for what feature. To summarise, the implementation chapter is about what was happening in the background. The evidence to support this can be in the form of a small screenshot of the code used to implement a key feature.

A section in the implementation chapter (probably at the end) - **Extra technical complexity/Functionality:** It has to be clear **how** you implemented the extra complexity and what the extra features are (if you have any). You need to signpost anything you believe is extra clearly in the report (Implementation chapter). The markers will not guess it by looking at the application. They must read it and identify it in the application in order to decide how complex they believe it is.

# Evaluation Chapter

What type of testing? Is there a test table? A little more explanation, for example, which type of testing is involved (black box, white box) unit testing ?....

# Conclusion Chapter:

Summarises what you did and future works.

# Team project Report - Approach for solving the problem: Software development methodology

Many usually lose their two marks for this section of the marking criteria "Has the team analysed the problem, and devised a suitable approach for solving the problem? ". It is important to know about software development methodologies and even crucial to discuss them in your report/dissertation. So in your report, you will discuss/compare a couple or 3 or more and indicate  which approach (es) you chose and why.

# How can we work well together, organise our team or share the tasks given our different skills?

 Organise roles/tasks after a clear understanding of project brief and requirements through continuous discussion and negotiation Software development is primarily about **communication** and sharing ideas to work towards a common goal. You need to spend time together collaborating. On top of your regular team project supervision slot on Fridays, you will need to arrange other meetings. It is all about communication, collaboration and negotiation. You could decide to use a different platform to meet. But remember that we will only look at your chat on Teams or emails as evidence should this be required.

 Not everyone will have programming skills or strong programming skills. If all members of a team are lucky to have prior programming skills, excellent. They can decide how and what features they will all code.

 If the team is of mixed abilities, make sure you use them. Remember that report writing is equally important as development. In the past, some teams have divided the team into programmers and report writers. Of course, before you even start coding (implementation), everyone should be involved in the planning phase. It means that everyone can do a bit, such as reviewing existing applications and literature on the importance of e-vehicles, etc., agree on the requirements, start designing the diagrams, etc. Teams can decide for all members to participate in planning, coding and report writing. In the past, some teams even had people just for testing. Some team members were paired for each task. These are just some ways I remember that teams did in the previous years. The bottom line is that it is all up to you. You have to discuss and negotiate amongst yourselves how you go about it. It is all about communication, collaboration and negotiation.

 Helping each other for the common good is very important!

# A teammate is not communicating and/or not contributing

 If a student is not communicating once, try using email and teams again. If the issue persists, let me know.

 If a student is not pulling their weight (not contributing) and not engaging with other team members to seek help when he is stuck somewhere (lack of communication and collaboration), they should know that this will be reported because they need to fill a contribution file where each member will list their contributions. It will affect the student's team project grade. **Notes**: we use contributions files when we notice some unusual delta scores and when there are problems. However, there could be a different outcome if there is evidence that the student has sought help from the others to help solve a problem together and the other members have all refused. So we deal with this on a case-by-case basis.

 In a team with one dominant person, that person should learn to listen to others and seek others' opinions. It is not individual work; it is a team-building exercise.

# No, you cannot work on a team project alone - you will not get a grade!!!!

In the past two years, a couple of people have attempted to submit their own reports and develop their own applications because

a) one of them did not get along with their teammates (it should be about communication, negotiation and respect for each other);

b) for the other one, what the team was working on was of poor quality, and that student had more skills (share the tasks, help each other to sort it out instead, become the team leader and support the others)

Obviously, these two students did not get any marks on Team Project because the clue is in the name **"Team"** project. Not "Individual" project! You will get the chance to shine alone during your summer project.

# ER diagrams

if you are struggling to understand what these are, watch this short video. <https://www.youtube.com/watch?v=QpdhBUYk7Kk>