4. Method Selection and Planning

A. Software Engineering

Methods and life cycles

- As a team we will be working via a spiral hybrid lifecycle, it was thought to be the most advantageous to introduce this model as it combines the idea of iterative development with the systematic, controlled aspects of the Waterfall model [1]. This is because it will minimise numerous risks (such as going beyond time limits, bugs, and requirements changing depending on customer specifications because these risks are explicit in the spiral nature).
- This lifecycle allows our team to iteratively develop the whole project; most likely in mini-increments to add functionality, this is very beneficial to creating a positive outcome as the project is constantly being updated and improved whilst keeping the solid basis of previous cycles. Following these cycles will lead to ideal software development. The Waterfall aspect also clearly defines the points of the project and sets out the team's clear goals. The hybrid approach will be both useful for the application design, planning and requirements as well as, development and testing.
- We will use the Agile methodology to develop our software but this does require the team to have good real-time communication and adaption to volatile requirements however, this group has from the outset had the ability and skills to do this effectively but also this method is applicable to this project because it focuses on reducing process overheads and documentation and on incremental software delivery.
- This method is developed in a series of increments and end-users and other system stakeholders are involved in specifying and evaluating each increment.[2] Which should result in an effective project solution with significant stakeholder input leading to a project which satisfies requirements. More specifically, aspects of XP (Extreme Programming) will be used to the fullest extent possible to develop our software.
- This is because four main purposes of using this method are to account for: dynamically changing software requirements; risks, caused by fixed timescales; small collaborative development teams; and appropriate tests. All of the aspects to XP described must be considered and expected and thus XP is how our project will be developed.
- Also when implementation starts, the fact that pair programming will be introduced will result in expedient code with fewer bugs in addition XP fits well with our chosen lifecycle. (However, verisimilar to in-work experience, XP integration can be hard, so like many companies, we may adopt only the most relevant and appropriate practises).[3]

Collaborative tools

- The team is employing multiple tools to complete the project. The project demands collaboration between all the requirements and all team members so, the use of Google Drives allows synchronous, asynchronous, distributed and in-person meetings thus providing the ultimate flexibility and the opportunity for producing the best project possible.
- On top of this, we are using GitHub for the joint implementation of the project and to input our application of the research and requirements elicited in order to create the final product. The java project will collaboratively and constantly be available to team members. So GitHub was an obvious and effective choice in this regard. This is

extremely beneficial as the use of this collaborative tool means we can keep different parts of the project separate but still use the same system meaning they are well organised, effectively managed, easily accessible and navigable by all team members.

- It was decided that GitHub Pages would be used for website creation and maintenance. The focus on end-user performance was ideal for what was required as it simply but effectively displays our project.
- IntelliJ and collaboration aspects of the IDE will be used for coding. It is clean, helpful and maximises developers productivity.
- Discord was also employed to coordinate and help discuss our project virtually and compartmentalise aspects of the project into different channels, which helps to focus on specific parts. Also as different calls can be created smaller groups can collaborate whilst still being in the same server allowing for competent use of the system to advance individual components and then en masse review our progress.

Fitness of tools (comparison to alternatives)

- Google drive was an obvious choice as it provided a slick and intuitive way to store our documentation in a freely accessible and collaborative way. Not only this, it has 15 GB storage capacity which will be more than enough for our project (a clear winner when compared to others e.g. dropbox). Google Drive syncs across multiple devices and operating systems so team members can always collaborate no matter the device (i.e. members can use iOS tablets and Windows laptops to work together on the same system).
- In GitHub for example, version control is extremely useful. This aspect means tracking changes is easy and identifiable, which is helpful when retrospectively viewing work (meaning asynchronous collaboration doesn’t cause redundancy and work can be clearly documented, removing confusion). Also, in certain circumstances, we may need to revert to a previous version, by using this software this can be done extremely easily which ultimately saves a lot of time. Other platforms were available (Bitbucket, TaraVault etc.) but GitHub for many reasons (especially the above) moreover, it is now the most widely used software by developers so, is familiar to the team furthermore, because it is a trusted cloud service provider with security the software element of our project is secure.
- GitHub Pages was compared against others (including Vercel etc.) and the comparison concluded that it would be the most suitable. Some benefits for example are that the website can be changed by team members easily, commits are made with version control, instant version rollback as well as having all other key features in similar tools allowing for effective updation and maintenance of the website.
- Similarly, the team is using IntelliJ to code the game and this software allows for integrated git version control allowing for storing versions if there is ever a need to revert. But it also includes code completion by context analyzing, debugging facilities etc. and has been ranked amongst the best IDEs for java so it would seem to be the optimum tool to use to produce our project.
- One of the overwhelming benefits of Discord is its familiarity. There would be no additional benefits that would help the project from competitor applications (slack etc.) and any other software would likely add superfluous and unnecessary content. The advantages already described (collaboration and designation) are core features that make Discord suitable.

B. Team Organisation

- The team approach to organisation is very methodical and focuses on the acceleration of output without loss of work integrity or quality. Explaining this further gives an in-depth view of the structure: first, the steady start begins, starting attentively was prudent as it lets the team figure out and gather all the information required before commencing. Because of this our team has not produced rushed or erroneous work.
- As the project continues work throughput increases at pace as all members know the tasks they have been assigned and are fully informed. This method allows adequate time for the whole team to review each other's work again adding to quality control and agreement. It is also important to expand on how this method works for the members of the team as well, in this regard members can more easily discuss work, prevent disagreements and be flexible in their work process (as time can be reallocated, see Gantt charts for effectual demonstrations of this).
- There will be allocated 2-hour (term-timed) weekly sessions which will make up a large proportion of review and discussion time but in addition to this, there are also Discord sessions (ranging between 30-90 minutes) to provide a supplementary period to complete and add additional content to the project.
- This way of organisation allows for both in-person and remote contact and along with the style of the two different meetings will define the purpose succinctly i.e. 2-hours will be largely content creation and review and the Discord one will be tying loose ends (which will benefit our team in the long run by keeping us within our time limits and well organised without long periods of unfinished work).
- The structure of our group means that not all members need to be present in meetings so their time can be used more productively and sessions become less cumbersome and more streamlined.
- Having the right balance of people for a specific section is important and the manager/managers of that section can get work done with the non-invasive oversight and outside perspective of at least one other member which engenders a work driven atmosphere but with the security of peer review from at least one other person both during and post writing.
- The hierarchy of the group is flexible and is based on a democratic and fair structure. This way of working may lead to less strong and decisive decision making however, for our team it is and will continue to be effective and due to this way of working a mix of ideas and perspectives are introduced to the project and everyone is as is far as possible pleased with outcomes that are made as a result of a whole group decision. Also, there are managers for different sections and a singular leader might not be able to make specific decisions for each part of the project so, having these managers make important decisions is crucial to the good progress of the project.
- We have created a logbook managed by one member which succinctly describes what has been achieved. This type of organisation helps to keep track of our progression along with the Gantt chart. This means that every week the group can start from the same point and the logbook acts as an external cognition aid to help the team members get on task quicker. All the described organisation above will efficiently and in an equitable way assist our project to run smoothly.

C. Systematic Plan

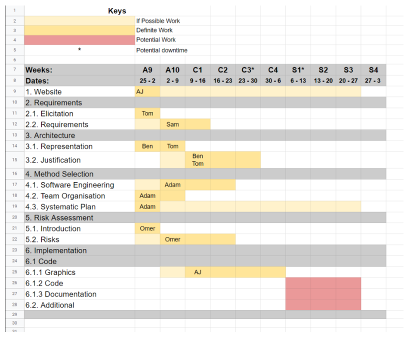
Knowing if a project can meet objectives on time is essential so having a systematic plan is required. As a group we will draw up a plan based on what should be delivered, when it should be delivered, and who will work on the development of the project deliverables. A plan-based approach requires a stable view of the development processes to accomplish this simple but effective planning method, a combination of written records and visual spreadsheets will be created which will adhere to the Agile methodology but also include the property of visibility. This can and does work for small companies developing software products but like every other professional software development process, Agile development has to be managed so that the best use is made of the time and resources available.[4]

For specific dates and how project modules interact please refer to the weekly updated Gantt chart. These will include key tasks and their start/finish dates.

Key tasks overview (A-Autumn, C-Christmas, S-Spring):

- Website A9-S3
- Elicitation A9
- (Dependent on elicitation) Requirements A9-C1
- (Dependent on requirements) Representation A9-A10
- (Dependent on representation) Justification A10-C3
- Software Engineering A9-C2
- Team Organisation A9-A10
- Systematic Plan A9-S3
- Risk Assessment A9-C2
- (Partially dependent on elicitation and requirements) Graphics A10- C4
- (Dependent of architecture i.e. representation) Code S1-S3
- Documentation S1-S3

These dates are subject to change (see Gantt charts for a full breakdown and changes).
An example of which is shown below (First week):



Keys	
▨	If Possible Work
🟨	Definite Work
🟥	Potential Work
*	Potential downtime

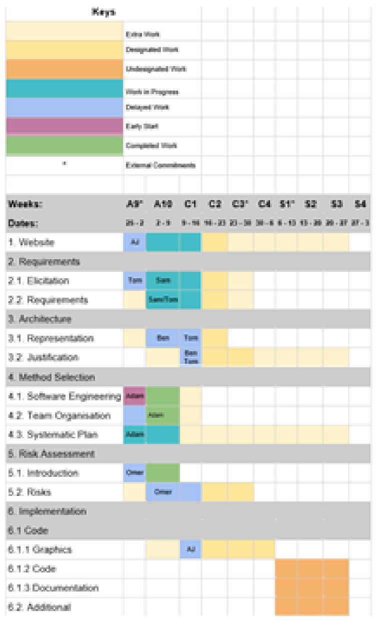
Weeks:	A9	A10	C1	C2	C3*	C4	S1*	S2	S3	S4
Dates:	25 - 2	2 - 9	9 - 16	16 - 23	23 - 30	30 - 6	6 - 13	13 - 20	20 - 27	27 - 3
1. Website	AJ	▨	▨	▨	▨	▨	▨	▨	▨	
2. Requirements										
2.1. Elicitation	Tom									
2.2. Requirements	▨	Sam	🟨							
3. Architecture										
3.1. Representation	Ben	Tom								
3.2. Justification		▨	Ben Tom	🟨	🟨					
4. Method Selection										
4.1. Software Engineering	▨	Adam	🟨	🟨						
4.2. Team Organisation	Adam	🟨								
4.3. Systematic Plan	Adam	▨	▨	▨	▨	▨	▨	▨	▨	
5. Risk Assessment										
5.1. Introduction	Omer									
5.2. Risks	▨	Omer	🟨	🟨						
6. Implementation										
6.1 Code										
6.1.1 Graphics		▨	AJ	🟨	🟨	🟨				
6.1.2 Code							🟥	🟥	🟥	
6.1.3 Documentation							🟥	🟥	🟥	
6.2. Additional							🟥	🟥	🟥	

The plan for the project requires cascading segments (dependencies) which will be completed by allocated team members but no dead space will be included as the project is constantly being worked on. These dependencies, for example, requirements are a prerequisite for class creation etc. in architecture, are done systematically and logically allowing for the best use of the available time.

As for priority, the current plan is equality, simply to achieve the best quality all parts are required, of course, mark distribution will play a part especially if a component is running behind.

We have set an ambitious and firm timetable but this is to ensure that the project continues to proceed at speed however, this allows for unexpected and unforeseen events because it has flexibility — of course, strict deadlines are still enforced to complete the module on time.

The Gantt chart has been developed to create a more systematic plan. This addition of more characteristics to the key indicates a more thought-through approach and helps to plan our time more effectively.



Keys	
🟨	Extra Work
🟡	Designated Work
🟧	Undesignated Work
🟦	Work in Progress
🔷	Delayed Work
🟪	Early Start
🟩	Completed Work
*	External Commitments

Weeks:	A9*	A10	C1	C2	C3*	C4	S1*	S2	S3	S4
Dates:	25 - 2	2 - 9	9 - 16	16 - 23	23 - 30	30 - 6	6 - 13	13 - 20	20 - 27	27 - 3
1. Website	🔷 AJ	🟦	🟦	🟡	🟨	🟨	🟨	🟨	🟨	
2. Requirements										
2.1. Elicitation	🔷 Tom	🟦 Sam	🟦	🟡	🟨					
2.2. Requirements	🟨	🟦 Sam/Tom	🟦	🟡	🟨					
3. Architecture										
3.1. Representation	🟨	🔷 Ben	🔷 Tom	🟡						
3.2. Justification		🟨	🔷 Ben Tom	🟡	🟡	🟨	🟨	🟨	🟨	
4. Method Selection										
4.1. Software Engineering	🟪 Adam	🟩	🟨							
4.2. Team Organisation	🔷	🟩 Adam	🟨							
4.3. Systematic Plan	🟦 Adam	🟦	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
5. Risk Assessment										
5.1. Introduction	🔷 Omer	🟩								
5.2. Risks	🟨	🔷 Omer	🔷	🟡	🟡					
6. Implementation										
6.1 Code										
6.1.1 Graphics		🟨	🔷 AJ	🟡	🟡	🟡				
6.1.2 Code							🟧	🟧	🟧	
6.1.3 Documentation							🟧	🟧	🟧	
6.2. Additional							🟧	🟧	🟧	

An advanced Gantt chart is better for our team than a standard chart as the extra information provided removes any ambiguity and streamlines planning.

Key is as below:

- Extra work - Additional work for extra improvement.
- Designated work - Future work timetabled.
- Undesignated work - Timetabled without designation.
- Work in Progress - Work started and will continue.
- Delayed Work - Due to dependency/commitments
- Early Start - Indicates work started before scheduled.
- Completed Work - Finished section.
- External Commitments (*) - Limited time/Restrictions etc. due to external factors.

Along with the Gantt charts, the logbook can be used retrospectively to plan future sessions effectively and systematically as it can be used as part of a methodical scheme to work out what needs to be done based on what has previously been done (also helping to explain and work out dependencies and keep to Agile methods).

Formatted with the same nomenclature of weeks and style of the Gantt chart, the addition of this document can be used hand in hand with other planning material to expand in a written way to provide descriptions of meetings and sessions acting as external cognition making it easier to see what has been done (even partially) and what needs to be done (described in more detail than a Gantt chart could provide). [5]

(Please refer to the logbook pdf [5] on the website for reference via the organisation dropdown).

References

1. Tutorialspoint.com. n.d. *SDLC - Spiral Model*. [online] Available at: <https://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm#:~:text=The%20spiral%20model%20combines%20the%20idea%20of%20iterative,with%20a%20very%20high%20emphasis%20on%20risk%20analysis.> [Accessed December 2021].

2. Sommerville, Ian. *Software Engineering, EBook, Global Edition*, Pearson Education, Limited, 2016. *ProQuest Ebook Central*, <http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=5185655> [Accessed December 2021].

3. Sommerville, Ian. *Software Engineering, EBook, Global Edition*, Pearson Education, Limited, 2016. *ProQuest Ebook Central*, <http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=5185655> [Accessed December 2021].

4. Sommerville, Ian. *Software Engineering, EBook, Global Edition*, Pearson Education, Limited, 2016. *ProQuest Ebook Central*, <http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=5185655> [Accessed December 2021].

5. Aj141299.github.io. 2022. *Home*. [online] Available at: <https://aj141299.github.io/YorKorsairs/index.html> [Accessed 31 January 2022].