Method selection and planning

The method decided by the team is a plan driven method, more specifically the object orientated methods which entailed using pre-existing models. To begin with, once we had conducted our user interview and using that created our user requirements, we knew method planning would be for us.

We started by using IMB's rational unified process (RUP) resulting in an incremental approach to the development of our game. We first started by going through a series of user stories which were written from the users perspective. Whilst going through the stories we made it a key priority to focus on not using computer science terms allowing us to fully focus on understanding what is needed. As a result there were many things that were left unstated however this was useful to get ideas flowing. The stories were also split into two distinct categories one of which being the "normal" cases in which things went well. These stories were beneficial to the design process as they allowed us to imagine what a polished game would look like and in turn allowed us to think about what could improve things. In addition we also had another type of stories, those being "exceptional" cases in which things indeed went wrong. This was useful as well as it allowed us to focus on what went wrong and in turn confirmed many of the entries on the risk assessment as well as creating new entries. This thoroughly improved the detection of potential bugs early on and it also gave us a good idea of what to look out for. From these stories we were then able to create sequence diagrams in which we laid out a visual representation of some of the specifics. The interaction between objects over time helped in understanding the design needed and it also aided in the documentation of the ideal system behaviour. This documentation served as guard rails and helped further down the line.

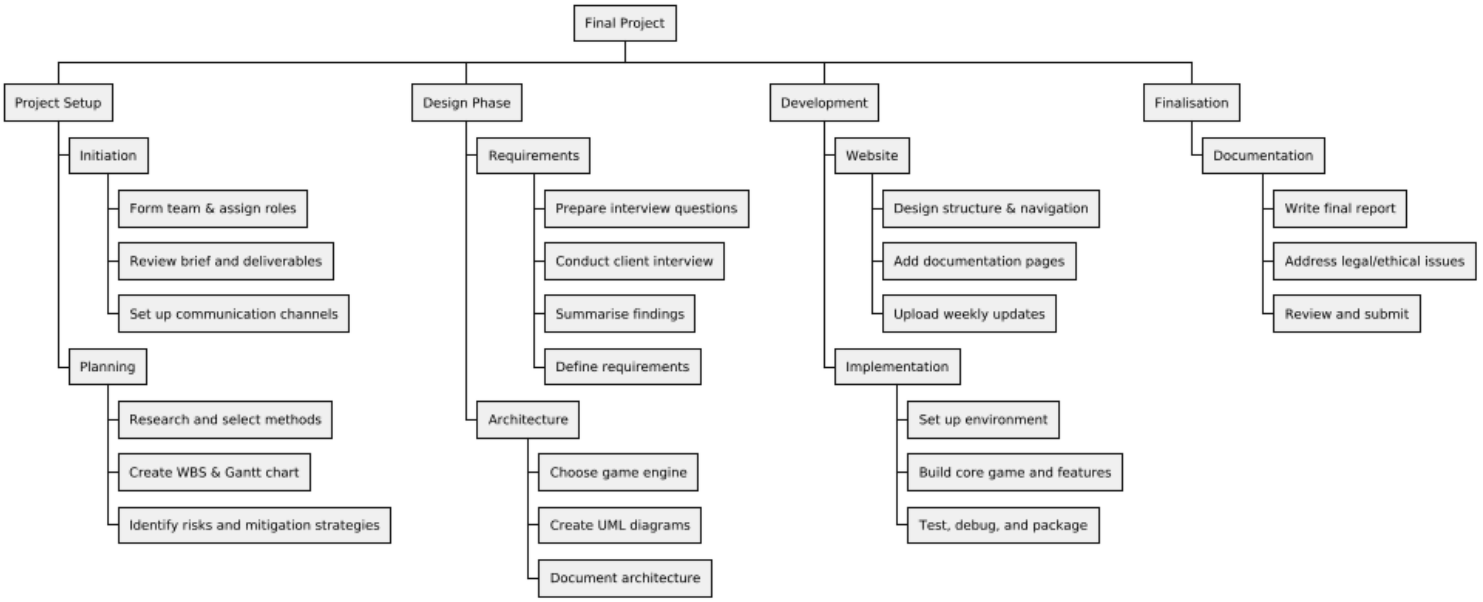
By using RUP we were able to start the first half of Model-Driven Development where we used the sequence diagram to come up with more abstract models. The process was a difficult process and resulted in multiple iterative designs of these models - which is something we expect to carry on through to the second half of the project. We came up with these models and designed them using UML (Unified Modeling Language) designing the objects in our domain-specific language (DSL). This DSL was Java and as a result we built the models to have methods and attributes.

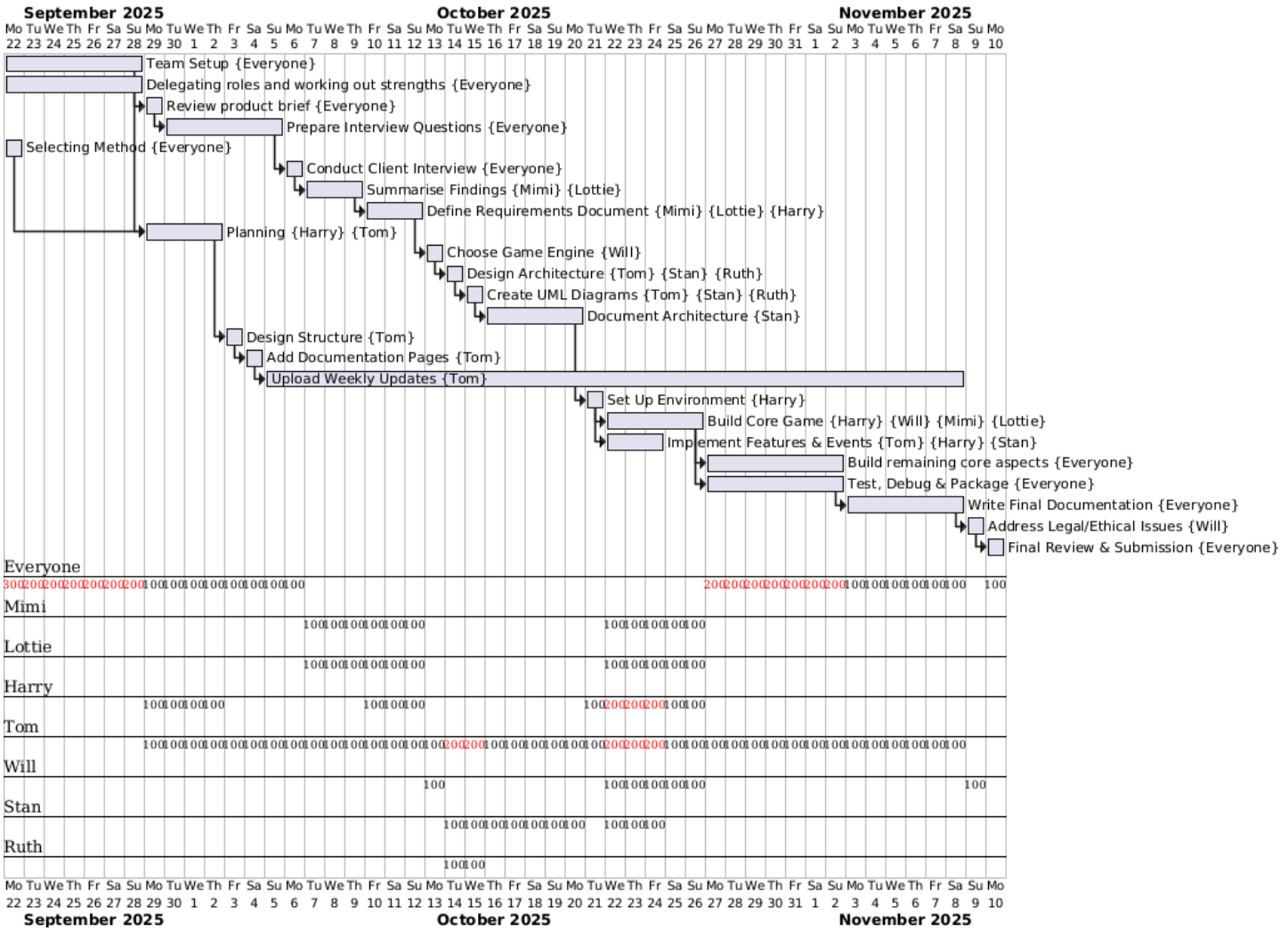
The tools we used were different types of graphs and diagrams from the plantUML library, something which I would say was a good fit for our chosen mode. To begin with, plantUML allowed us to directly create sequence diagrams which allowed us to get a visual representation of the two types of cases. In addition it also allowed us to share the created sequence diagrams allowing us to work without being together. In addition, plantUML also allowed for us to get an overall feel for what the project would look like architecturally as well as getting an idea of how we would be able to split up the work into fairly manageable sections. The use of the tool to map out the steps needed to be taken to build the project acted as a backbone to future aspects of the project, more specifically its planning.

In terms of other methods considered the group took great liberty to assess the pros as well as the cons of other methods. The first alternative we assessed was the agile method due to its current day relevance as well as its popularity in the industry. We agreed it was a smart way to approach the project with the short iteration cycles, in which there was incremental

delivery of value to the customer. The rapid development and lack of documentation was appealing as it would allow us to constantly focus on fitting the project to the customer. The philosophy that humans have changing requirements was something we liked. Although due to the nature of the group project we viewed it wouldn't be viable as the product brief was for the most part given to us and in turn would result in very little changes to the brief. In addition, although areas of the agile manifesto were very appealing, in particular the view of working software over comprehensive documentation, this would not allow to complete the assessment fully. In addition we also predicted that we would not be able to contact our customer as often as needed to complete the project using the agile method. This is something that was a big limiter for us. We thought there would be a struggle getting face to face meetings with our customer, something which is stated as a key feature of the agile methodology. We also thought the need for constant group meetings was something which we viewed as invisible due to the busy and unpredictable nature of people's schedules as well as everyone having different timetables. As a result we viewed that the aspect of teams needing to regularly reflect on how to become more effective, continuously tuning and adjusting behaviour was not possible for our team.

For the team organisation we ultimately organised around a key idea that splitting into groups is far more effective and beneficial which led to a breakdown in tasks and then assigning to those who were interested by the task as well as those that were free. To begin with the team assigned roles to everyone, something which we felt was necessary in order to avoid members being overworked or overwhelmed. These roles later transferred into responsibilities, which was something that came about naturally. The roles were dictated on what people thought were their strengths although were later finalised further along the development process as people either wanted to explore new roles and grow their skills. We first made it a priority to break down the overall brief and requirements into smaller sections which were handed off to people based on their roles. The roles also provided the key structure needed for group meetings as they resulted in people being able to be more productive. Communication as a team was a big thing for us as we viewed it as the fundamental factor that would contribute to our success. This is as we know software engineering is a group based thing which in turn means that if we were unable to communicate ideas properly we wouldn't be able to achieve the collaborative nature needed to succeed. As it was such a priority for us we initially started the project by completing team building exercises. We also set up a group chat in order to stay connected, share ideas and review aspects of the project. In addition we decided to have a democratic decision structure which resulted in new ideas being voted upon before concrete implementation. This we felt was a fair way to take on new ideas allowing us to move in a direction that we were all comfortable with. As a result of this decision we often resulted in a combination of various ideas, the final additions to games were often something which was iteratively changed to incorporate all ideas. This is something which we not only viewed as fair but it also helped to boost communication as it resulted in everyone not only feeling heard but happy with the decisions made. This was at least for the most part as it favoured the majority. Finally when it came to planning and workflow the planning aspect was something that was iterative and oftentimes influenced by the workflow. The team focused on constantly updating and changing the plan as the addition of new features resulted in peoples workflow changing to work on other aspects. Moreover illness oftentimes resulted in people changing tasks and changing the plan.


Final Project
Project Setup
Initiation
Form team & assign roles
Review brief and deliverables
Set up communication channels
Planning
Research and select methods
Create WBS & Gantt chart
Identify risks and mitigation strategies
Design Phase
Requirements
Prepare interview questions
Conduct client interview
Summarise findings
Define requirements
Architecture
Choose game engine
Create UML diagrams
Document architecture
Development
Website
Design structure & navigation
Add documentation pages
Upload weekly updates
Implementation
Set up environment
Build core game and features
Test, debug, and package
Finalisation
Documentation
Write final report
Address legal/ethical issues
Review and submit


September 2025
October 2025
November 2025
Team Setup {Everyone}
Delegating roles and working out strengths {Everyone}
Review product brief {Everyone}
Prepare Interview Questions {Everyone}
Selecting Method {Everyone}
Conduct Client Interview {Everyone}
Summarise Findings {Mimi} {Lottie}
Define Requirements Document {Mimi} {Lottie} {Harry}
Planning {Harry} {Tom}
Choose Game Engine {Will}
Design Architecture {Tom} {Stan} {Ruth}
Create UML Diagrams {Tom} {Stan} {Ruth}
Document Architecture {Stan}
Design Structure {Tom}
Add Documentation Pages {Tom}
Upload Weekly Updates {Tom}
Set Up Environment {Harry}
Build Core Game {Harry} {Will} {Mimi} {Lottie}
Implement Features & Events {Tom} {Harry} {Stan}
Build remaining core aspects {Everyone}
Test, Debug & Package {Everyone}
Write Final Documentation {Everyone}
Address Legal/Ethical Issues {Will}
Final Review & Submission {Everyone}
Everyone
Mimi
Lottie
Harry
Tom
Will
Stan
Ruth

Initially the plan started with a similar layout to that of the assessment paper with a clear layout for the project setup, design development as well as implementation and review. This clear layout was very much linear and something which was structured as we had not yet built or undergone much of the process to develop this game. We built a work breakdown structure as well as a gantt chart similar to the ones above and both were built to adhere to a clear structure. At first they were set up as to outline estimated time for things to finish with idyllic timing as well as dependencies which were very much linear and far easier to follow. There was very much a consensus that each stage was supposed to follow on from the last.

Whilst the initial plan was useful as it set the groundwork for laying out a more precise plan and one that slowly shaped into the current one, it was clear to see that the original plan would not hold. To begin with coding tasks took far longer than expected and were started perhaps later than expected resulting in people having to work under perhaps more stressful conditions. Furthermore it also reduced any testing coming much later than expected and perhaps not thorough. Moreover the parallel work started much earlier than expected, this is because some people were tasked with documentation whilst others were perhaps tasked with research. In addition, in weeks where there was less to do documentation was the focus and as a result the parallel work was perhaps something which we did not expect to take such an effect as it did to the plan. The addition of new tasks also impacted the original plan as this is because once building the core game we realised we were missing some not so apparent key aspects to the brief something which we did not realise till the first review of the game. Furthermore some tasks were shortened as they initially deemed to take longer or the team developed more efficient ways of completing the task resulting in a redshift of the plan. Moreover the constriction of time also resulted in a shift in the plan as the team shifted away from focusing on the polish of the game to the functionality, as it was more important for the game to be functionally sound rather than perfectly polished. Moreover, unexpected changes to the game structure resulted in a change of the plan. This is because certain aspects required more time to debug and fully implement and as a result caused things to shift back in regards to when they were supposed to be finished. Furthermore when building the game, it became apparent that there were far more dependencies than expected resulting in a shift in the dependencies. All the above is reflected in the selected weekly snapshots we decided to keep uploaded on our website.

Overall despite the plan not sticking to the original outline of how things were supposed to go the process taught us many valuable lessons. To begin with, the weekly snapshots and updates helped track the progress, allowing us to manage resources as best as we could, as well as keeping it looking more realistic. It also taught us how to stay flexible and adapt instead of following a rigid plan, which is something we valued. Furthermore the importance of the weekly snapshots allowed us to stay aware of the looming deadline keeping the team focused and consistent. In addition adjusting the dependencies helped us manage delays and resulted in better team cohesion. All in all despite the constant changes, the changes to the plan were impactful and the development of it resulted in a much more realistic plan that served as a reminder of deadlines.