Method selection and planning

The team decided to use a plan-driven model approach, specifically an object orientated method which utilised pre-existing models. After conducting our user interview and developing our user requirements, we determined that a structured, plan-driven methodology 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. Initially we went 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 also proved useful 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.

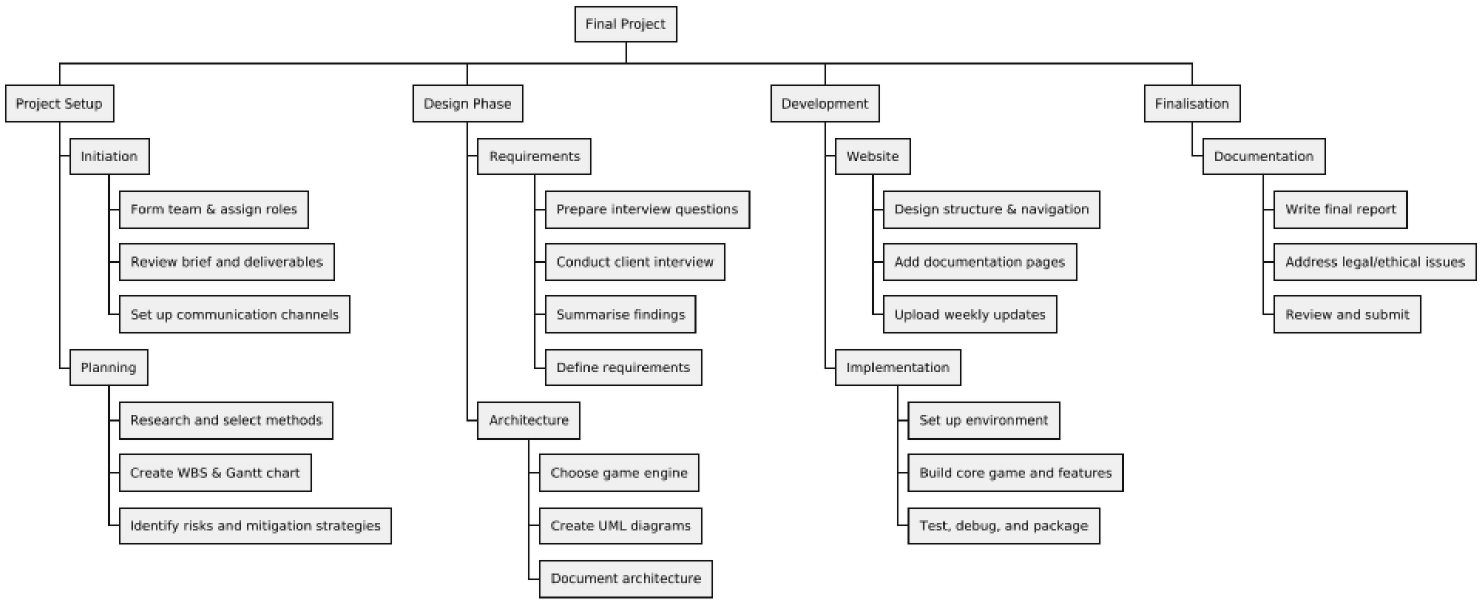
By using RUP we were able to start the first half of Model-Driven Development where we used the sequence diagram to come up with more abstract models. The process was difficult and resulted in multiple iterative designs of these models - which is something we expect to carry on through to the second half of the project. We came up with and designed these models 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.

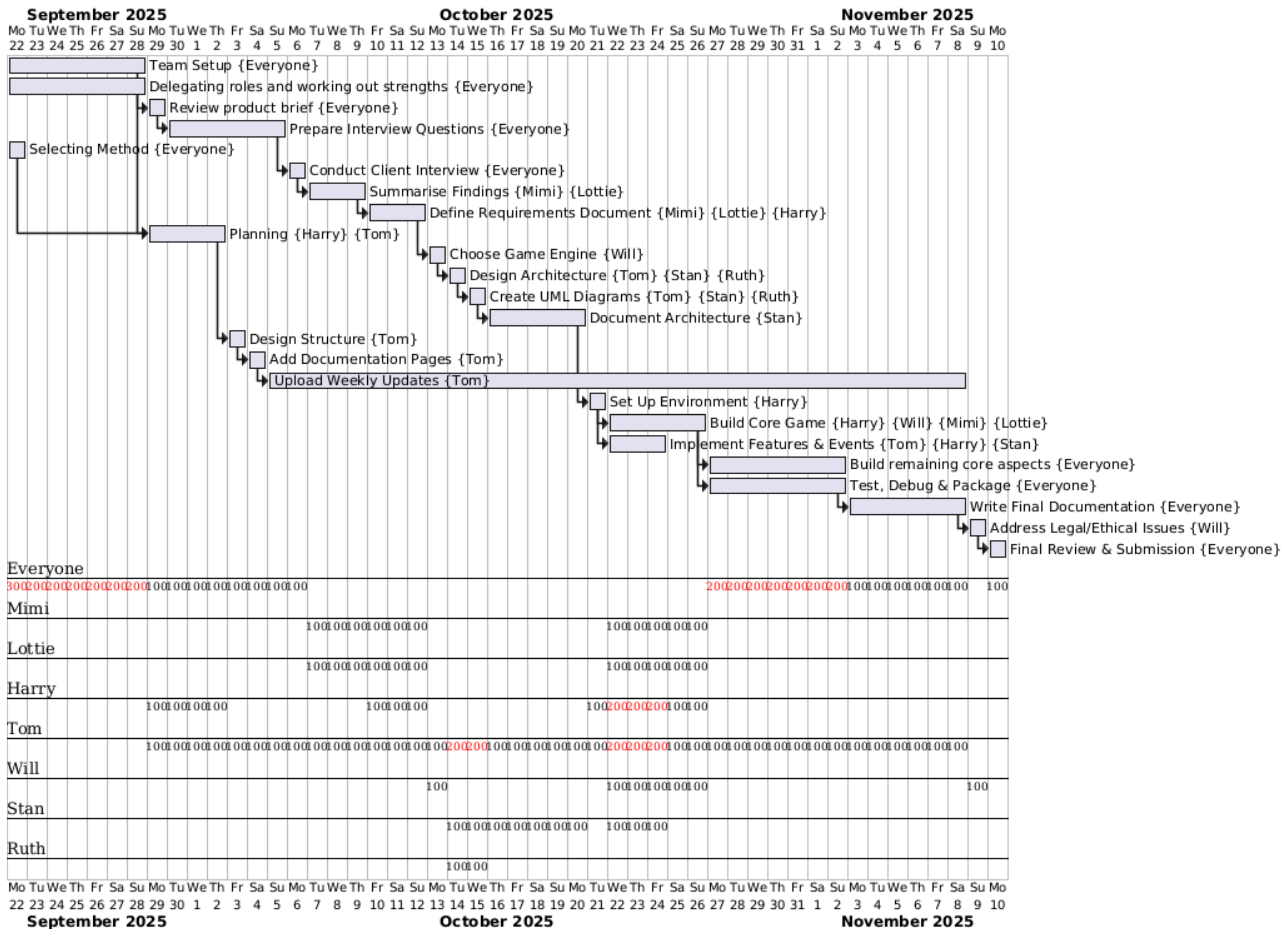
The tools we used were different types of graphs and diagrams from the plantUML library, something which was a good fit for our chosen method. To begin with, plantUML allowed us to directly create sequence diagrams which allowed us to get a visual representation of the two types of cases. In addition it also allowed us to share the created sequence diagrams allowing us to work without being together. In addition, plantUML also allowed us to get an overall understanding of the project's architecture and to identify how we could divide the work into manageable sections. The use of this tool to map out the steps required to build the project served as a foundation for future stages, particularly the planning phase.

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. We also appreciated the underlying philosophy of Agile - that human needs and requirements naturally evolve over time. Although due to the nature of the group project, we viewed it wouldn't be viable as the project's key requirements were already set, allowing for very limited changes. Nonetheless, although the Agile manifesto's emphasis on working software rather than extensive documentation was compelling, adopting this methodology would've prevented us from meeting all aspects of the assessment criteria. In addition, we anticipated that we would not be able to communicate with our customer frequently enough to successfully implement the Agile method, which proved to be a significant limitation. Moreover, we thought there would be a struggle in 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 impractical due to the busy and unpredictable nature of the team'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 decided that splitting into smaller groups would be far more effective and beneficial. This approach allowed us to break down tasks and assign them based on interest and availability. 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. While roles were originally dictated based on each members' perceived strengths, they were later revised during the development process as individuals sought to broaden their experience and enhance their skills. After delegating roles we then 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. Establishing roles helped create a clear framework for group meetings, which in turn improved overall productivity.

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 belief arose from the understanding that software engineering is inherently a collaborative endeavour; without effective communication, we would be unable to achieve the teamwork necessary for success. As communication was such a priority for the team, we initially started the project by completing team building exercises. We also set up communication channels, 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 approach provided a fairer process for evaluating new ideas, ensuring that we moved forwards in a direction everyone supported. This process often led to final game features that combined multiple ideas, adjusted iteratively to include everyone. Finally, the planning process was iterative and frequently influenced by the team's workflow . The team prioritized constantly updating the plan, as the addition of new features often shifted members' focus to different tasks. Additionally, unforeseen absences, such as illness, frequently required adjustments to both individual responsibilities and the overall 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. Initially the plans were structured to outline estimated completion times with idealised schedules and clearly defined dependencies, which were largely linear and easy to follow. There was a strong consensus that each stage was supposed to follow sequentially from the previous one.

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 was due to some people being 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, something which we did not realise till the first review of the game. Furthermore some tasks were shortened as they were initially deemed to take longer or the team developed more efficient ways of completing the task resulting in a shift 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. Additionally, unexpected changes to the game structure resulted in a change of the plan. This could be attributed to certain aspects requiring more time to debug and fully implement and as a result caused deadlines to shift back. Furthermore during game development, it became apparent that there were far more dependencies than expected, necessitating a reassessment and adjustment of those dependencies. These changes and adjustments are documented in the weekly snapshots we chose to upload and maintain on our website.

Overall despite the plan not adhering strictly to the original outline, the process taught us many valuable lessons. To begin with, the weekly snapshots and updates helped track progress, enabling us to manage resources effectively while keeping the project plan 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. Overall, while the plan underwent continuous changes, these adjustments proved valuable, resulting in a more practical timeline that reinforced key deadlines.