METHOD SELECTION AND PLANNING

Cohort 1 group 11

Freddie Aberdeen
Mutaz Ghandour
James Given
Jasper Owen
Jungwan Park
Joe Reece
Ivan Shestakov

Software Engineering Methods:

Outline and Justification of Methods:

The team adopted an Agile software development methodology, specifically implementing the Scrum framework. Agile was chosen because it supports iterative development, continuous feedback, and adaptive planning - all of which align with the project's need for flexibility and ongoing refinement. The project involved multiple interdependent components so a rigid, plan-driven method would have limited our ability to work efficiently.

Scrum provided a structured yet flexible framework. We divided development into weekly sprints, each beginning with a sprint review of the previous week, and ending with sprint planning for the following week. This cycle enabled:

- Incremental development: delivering features in small, modular units to facilitate maintainability and future expansion
- Frequent feedback loops: improving quality and alignment with customer expectations
- Cross-functional collaboration: allowing team members to integrate and test their contributions continuously

Development and Collaboration Tools:

To support our Agile methodology, the team used several tools to streamline collaboration, task management, and development:

GitHub - Version Control:

- Required a version control system for saving previous versions, tracking changes and facilitating collaborative development, chose GitHub because:
    - Allowed team members to collaborate safely and maintain a history of all changes
    - Supports distributed development, branching, and merging workflows.
    - Pull requests enabled code review and traceability, aligning with Scrum's collaborative ethos.
- Alternatives: GitLab was considered, however GitHub was preferred for it's integration with other tools and familiarity among team members.

GitHub Issues - Task Management:

- Required a task management tool to keep track of the status of tasks, chose GitHub Issues because:
    - Provided a clear, visual overview of what tasks still needed to be completed and who was responsible for them.
    - Tasks could be easily assigned and prioritized.
- Alternatives: Jira was considered, we deemed it overly complex for our needs.
- Assigned weekly tasks to each team member and kept the development workflow organised.

Visual Studio Code - Code Editing and Debugging:

- Justification: Lightweight IDE with integrated Git support and extensions for collaborative development, aligning with Agile's emphasis on developer productivity
- Alternatives: IntelliJ was considered but required more configuration and was less consistent across all team member's systems.

Google Docs - Documentation:

- Justification: We all have prior experience using Google Docs, including in Assessment 1 and it is easy to share access to documentation by sharing documents
- Alternatives: While we could have used other software such as shared Word Documents in OneDrive, we were all more familiar with Google Docs.

WhatsApp Community - Communication:

- Justification: We used a WhatsApp Community to communicate with each other through the use of multiple group chats,share important information and adjust the schedule of meetings.
- Alternatives: We could have used a Discord server for the same purpose but we decided to use WhatsApp instead due to personal preference

Team Organisation:

The team approached team organisation by dividing the project into two main groups - documentation and coding - and then further subdividing tasks within each group. This was done because there was a lot of work to do in both documentation, where the majority of our marks came from; and the coding of our game, which was the main source of our documentation. Therefore to manage the project we divided into groups, with some crossover to ensure that the documentation and code were consistent with each other.

This approach to team organisation was effective for this project as it aligned closely with agile principles: promoting autonomy, transparency, and iterative collaboration. Since we clearly divided the work, each member was able to work independently on their respective components, which greatly helped workflow efficiency as people were able to work whenever was convenient for them. In order to maintain consistency, we conducted two weekly meetings which ensured that each member's work was cleanly integrated into the project and that work was being completed on schedule.

Plan:

When it came to planning our work on a weekly basis we used Gantt charts to make sure everything stayed on track. Utilising this approach for planning allows us to easily identify dependencies as with most Gantt charts the tasks to the right are dependent on completion of the tasks to the left. These gantt charts were used to break down our keys weekly tasks into bite sized chunks that everyone related could work on.

Visual representations of the plan can be found on the team's website (https://escape-from-uni.github.io) in the form of Gantt charts.

Our broad objectives throughout the project, starting on 22nd September 2025 and ending on 12th January 2026 were:

- Create a University-Themed maze game
- Include 3 positive events that help the user
- Include 5 negative events that hinder the user
- Include 3 hidden events the user cannot initially see
- Include a 5-minute timer in the game
- Make it so the player gets a score at the end of the game where finishing earlier gives them a higher score
- Write all the documentation to go with our game
- Create a website to put our game and documentation on

For the initial phase of game development, our goals that started on 22nd September 2025 and ended on 10th November 2026 were:

- Create a University-Themed maze game
- Include a positive event that helps the user
- Include a negative event that hinders the user
- Include a hidden event the user cannot initially see
- Include a 5-minute timer in the game
- Make it so the player gets a score at the end of the game where finishing earlier gives them a higher score
- Write the initial drafts for the following documentation:
    - Architecture
    - Risk Assessment
    - Method Selection
    - Implementation
    - Requirements
- Put our work on our website

For the second phase of game development, our goals that started on 24th November 2025 and ended on 10th November 2026 were:

- Finish our game
- Include 2 more positive events that help the user
- Include 4 more negative events that hinder the user
- Include 2 more hidden events the user cannot initially see
- Include a leaderboard
- Include achievements that can be achieved by the user
- Write final drafts for the following documentation:
    - Architecture
    - Risk Assessment
    - Method Selection
    - Implementation
    - Requirements
    - Change report
    - Testing
    - User Evaluation
- Put our new and updated work on our website

Week 1: Team building and Method selection:

Key tasks:

- Research possible java game libraries
- Understand the Assessment requirements
- Assign roles within the team

Priorities:

- Evaluate each members skills

Dependencies:

- None

Week 2: Interview and Requirements development

Key tasks:

- Create interview questions
- Create a comprehensive set of requirements taken from the client
- Assign functional and non-functional requirements of the system

Priorities:

- The whole team must be knowledgeable about the requirements by the end of week

Dependencies:

- Interview performed earlier in the week
- The transcript created from the interview

Week 3: Initial designs and Risk Assessment

Key tasks:

- Brainstorm a design and plan the gameplay loop
- Create a Risk Register covering all pitfalls during the project

Priorities:

- Have an initial game loop and design the players experience

Dependencies:

- Design dependent on the requirements given by the client
- Risk assessment dependent on the team choosing roles and tools for the project

Week 4: Architecture and Implementation

Key tasks:

- Start base implementation of generic movement and controls
- Create a plan for the architecture and class systems of the game

Priorities:

- By the end of the week have a moving sprite on the screen
- Have a design of what the structure of the game will look like

Dependencies:

- Both dependent on having the base design of what the game should do

Week 5: Implementation

Key tasks:

- Implement a tile map system to allow a graphical map
- Implement object interaction for the events in the game
- Implement timer
- Start finalising documentation

Priorities:

- More implementation

Dependencies:

- Simple game implementation must be complete before this

Week 6: Documentation Review and Submission

Key tasks:

- Finish and review documentation
- Give the game some final polishing

Priorities:

- Submission by end of week

Dependencies:

- All documentation must be finished
- Game must be functional with its mechanics

In weeks 7 and 8 the group that created this project for Assessment 1 presented the project and chose a new project to work on. Then in week 9 we began to continue developing this project for Assessment 2.

Week 9: Preparing project for future development

Key tasks:

- Review code and prepare it for future development
- Make necessary editions to documentation

Priorities:

- Have code ready for development

Dependencies:

- Changes to Architecture document can only be made once the code has been reviewed and edited, in case any architectural changes are made

Week 10: Continuing development of the game

Key Tasks:

- Write Change report
- Prepare for User Evaluation
- Continue developing the code

Priorities:

- Be ready for User Evaluation by Tuesday

Dependancies:

- For the User Evaluation, Ethics fast track document must be filled in and submitted, and Informed Consent form and Information sheet must be prepared

Week 11: Preparing to continue development over holidays:

Key tasks:

- Continue game development
- Write User-Evaluation report
- Rewrite the Architecture Document
- Write Architecture changes in change report

Priorities:

- Prepare to add more events to the game

Dependancies:

- Adding Architecture to change report is dependant on there being changes to document

Christmas Holidays: Continue developing the game and the write up over the holidays:

Key tasks:

- Finish User evaluation write up
- Continue game development
- Start developing game tests
- Review Assessment 1 feedback
- Apply Assessment 1 feedback to our own work
- Start Software Testing Report
- Set up Continuous Integration infrastructure
- Start Continuous integration write up

Priorities:

- Start adding Tests
- Continue game development

Dependancies:

- For the Software Testing Report, tests had to be written and test data had to be collected
- We wanted to have our tests as part of the game before starting the Continuous Integration Report

Revision week: Finish developing the game

Key tasks:

- Finish documentation write-ups
- Finish game
- Finish adding tests
- Submit coursework

Priorities:

- Finish adding tests
- Finish developing the game

Dependancies:

- For us to submit, everything had to be ready to submit
- For us to finish adding tests the game had to be completed
- For us to finish the testing report all our tests had to be written and all tests we wanted to succeed should have succeeded.