Change Report

Cohort 3 Team 4

Kiran Kang

Hannah Rooke

Ben Slater

Abualhassan Alrady

Cassie Dalrymple

Charles MacLeod

Dash Ratcliffe

Harley Donger

Development tools:
The team will be using IntelliJ IDEs to develop in Java code, this means that the people who are writing the actual code will be able to collaborate with each other, sharing information about keybinds, extensions, and other capabilities streamlining the development process, rather than if each developer was using either other IDEs or just plain text editors. IntelliJ includes a rudimentary built-in formatter, that will for example indent the code correctly on save. The team members responsible for code development did not feel like using a third-party formatter would be useful, and it would just implement mild setup delays.

Code-specific collaboration:
For collaborating with code, the team will be using Git, and GitHub (as the platform to hold the code, discussions and revision history). The reason we have decided to use Git, is that it is an industry-standard way to keep track of version control, and see the progression of how the game code changes with the collaborators. There is an abundance of guides and documentation online on how to use Git, including all of its commands, etc.

From there, we have decided to (specified in the CONTRIBUTING.md file in the repository) use pull requests and branches (a mechanism that allows contributors to propose changes to a project through in their own isolated branches of code, which can then be reviewed and discussed before being merged into the main codebase) to have a structured workflow that allows each change to be isolated and examined by every collaborator before being merged into the main codebase. Each commit also follows conventional commit naming, such as “feat“, “fix”, “docs”.

There will also be a .gitignore file (a file which excludes certain files from being tracked by git, and therefore uploaded to the repository) to keep the repository clean and exclude any build logs, caches, etc that are not needed when someone is browsing the code.

Non-code collaboration:
One collaboration medium that is not for code, we have decided to use Google Docs and Google Drive. This decision was made with the main reason being familiarity with the tools - everyone on the team has used these tools before, and each of our university Google accounts has access to 2TB of storage space, which is plenty for this kind of work. We have split it into multiple files that we each have access to and can edit, and the main people working on each deliverable are responsible for working on the documents. Google Drive is simply used to hold all of these files that we work on.

We will also be using WhatsApp as the primary “informal” communication platform, and once again, the biggest factor drawing us to choose this platform was everyone’s familiarity with it, and how standardised across the world it is.

Project planning tools:
For project planning, we have also decided to use Github issues, they are not just for issues, but can have all forms of discussions. Alternative tools for this include Jira, Trello and Asana,

and we looked at them, and while they had free plans, either nobody or only a few people on the team ever used them, whereas most are familiar with Github, and keeping things to just one platform reduces overhead for the development and documentation teams.

Level development:

For the map development, to design and plan each of the levels for the game, we use a software called “Tiled”. It is a free, open source and full-featured level editor, that supports flexible object layers, is extensible with JavaScript, and is widely supported by many game development frameworks, such as Unity, Godot, as well as lots of languages like C, C#, Go, Java (what we are using for the game), Python, Rust, and many others.

We chose to use Tiled since it is more popular that other widely used and free alternatives, such as LDtx, but that is also a free and open source 2D level editor that is widely used and supported.

CI setup:

For continuous integration, we use Github Actions. A big reason for this once again is that it is integrated into the platform that we are already using for code hosting, issue tracking, discussions, etc. We have learned in one of the lectures how to set up basic Actions, so we will use it to run builds, and checks to make sure it compiles and runs on the platforms outlined in the requirements, get the compiled jar file, upload it as an artifact and create releases with the Jars attached. Some alternatives we considered to use were GitLab CI/CD (it is powerful and YAML-based, and great if the repo is on GitLab, but our repo is on Github and using it adds friction (mirrors/tokens) and an extra ui to manage) and self-hosted Jenkins (we would have full control but we would have to provision servers, plugins, backups, etc).

Definitions and change control:

To keep work predictable, an issue is Ready when it satisfies a requirement from the list of requirements derived from the initial (or subsequent) client meetings. A change is done when it is approved by the programming team, and CI is accepting on Linux, macOS and Windows and a JAR runs locally. If any issues arise, they are opened on GitHub issues to be addressed.

We implement the desktop game with Java 17, libGDX and Gradle. Java 17 is the mandated language level and provides a portable, cross-platform runtime. libGDX is a lightweight 2D engine with a mature LWJGL3 desktop backend, tiled-map support, text rendering and audio, which helps us target low-spec laptops and meet the client’s cross-platform requirement. Gradle offers fast builds as well. An alternative to Gradle is Maven, which is a robust build tool as well.

Team approach to organisation:

We operate without a formal leader. At the start we assigned work based on preference and skills (e.g., requirements, risk assessment, coding). GitHub is our single source of truth: all work flows through Issues and PRs with labels and assignees (no templates, no requirement IDs). The main branch is protected; merges for the game code go via reviewed PRs, while low-risk repository admin/docs changes (e.g., CONTRIBUTING.md) can go directly to main. Small decisions are made within the relevant sub-team (e.g., the two programmers) to avoid blocking others; big decisions (direction, scope, trade-offs) are discussed biweekly in a whole-team meeting with updates/demos and decided by vote. Day-to-day communication

and quick questions are handled asynchronously on WhatsApp.

This approach fits the team and project: everyone expressed satisfaction with the setup, it minimises ceremony while enforcing quality with things like branch protection, code review and visible ownership on Issues/PRs, and it is using tools familiar to everyone (GitHub, Google Docs/Drive, WhatsApp) to reduce overhead and suit mixed student schedules. Pairing in fragile areas and occasional role rotation mitigate single-point-of-failure risks highlighted in our risk assessment. If anyone needs history, we have a version history for each file through google docs and google drive, and for the code, there is a commit history in the repository.

Systematic plan:

Milestones and dates (A1 due Mon 10 Nov 2025):

- Iteration 1 (1–7 Oct): Risk register baseline; requirements skeleton; website scaffolded.
- Iteration 2 (8–14 Oct): Requirements expanded; architecture baseline. ● Iteration 3 (15–21 Oct): Architecture refined; risk register matured. ● Iteration 4 (22–28 Oct): Repo scaffold (Java 17, Gradle, libGDX); CI setup. ● Iteration 5 (29 Oct–4 Nov): Map pipeline (Tiled) and art; event system skeleton. ● Iteration 6 (5–10 Nov): Timer/pause/counters; polish, packaging, PDFs and website.

Key tasks with dates, priorities, dependencies:

T001 - Risk assessment and mitigation
- Start: 1 Oct • Finish: 28 Oct
- Dependencies: none • Output: Risk1.pdf, risk register (IDs, likelihood/impact, mitigation)

T002 - Requirements elicitation and specification
- Start: 2 Oct • Finish: 7 Nov
- Dependencies: none • Output: Req1.pdf with UR/FR/NFR IDs and acceptance criteria

T003 - Architecture (structural + behavioural, traceability)
- Start: 9 Oct • Finish: 7 Nov
- Dependencies: T002 (req IDs) • Output: Arch1.pdf with Doc

T004 - Tooling and CI (Java 17, Gradle, libGDX)
- Start: 22 Oct • Finish: 3 Nov
- Dependencies: none • Output: Initial code in the github repository

T005 - Map pipeline and art (Tiled + layers)
- Start: 29 Oct • Finish: 7 Nov
- Dependencies: T004 (project scaffold) • Output: maze assets and layers

T006 - Event system skeleton (triggers/effects)
- Start: 29 Oct • Finish: 4 Nov
- Dependencies: T005 (map loader) • Output: Triggering mechanism, effects interface

T007 - Implement required A1 events (1 negative, 1 positive, 1 hidden)
- Start: 30 Oct • Finish: 7 Nov
- Dependencies: T006 • Output: Three working events, counters incremented

T008 - Timer, pause and counters UI
- Start: 22 Oct • Finish: 5 Nov
- Dependencies: T004 (loop/input), T006 (event outcomes for counters) • Output: 5-minute timer with pause, counters overlay

T009 - Packaging
- Start: 1 Nov • Finish: 6 Nov
- Dependencies: T004, T008 • Output: Single executable JAR, validated on Windows/macOS/Linux

T010 - Accessibility and attribution
- Start: 1 Nov • Finish: 7 Nov
- Dependencies: T005–T008 • Output: Multimodal cues (text + visual/audio), Assets/ATTRIBUTION.md

T011 - Website and deliverables
- Start: 1 Oct • Finish: 9 - 10 Nov
- Dependencies: all • Output: Website with links to PDFs, JAR, repo; final PDFs (Req1.pdf, Arch1.pdf, Plan1.pdf, Risk1.pdf, Impl1.pdf)

Dependencies summary:

- T003 depends on T002 (traceability from requirements).
- T006 depends on T005 (map loader and layers).
- T007 depends on T006 (event framework).
- T008 depends on T004 (input/game loop) and T006 (event outcomes).
- T009 depends on T004 and T008 (build + artifact).
- T011 depends on all prior tasks.

How the plan evolved:

Our initial plan front-loaded discovery and documentation in the first two weeks, so things such as risk register, meeting with the client to create the requirements skeleton, a website scaffold. As requirements solidified, we produced a baseline architecture (structural first, then behavioural), and scheduled weekly snapshots to capture changes.

After scaffolding the libGDX project (Java 17 with Gradle), we learned that integrating Tiled maps and event triggers would take longer than estimated because map loading and collision layers needed more iteration. We pulled timer/input work forward and pushed the event-system skeleton back by one week. This change was recorded as a plan update in the next snapshot.

The requirements grew to include main menu and settings - we retained these in the spec for completeness but re-classified them as should/could for A1. We also tightened the wording of hidden events and added acceptance criteria, which improved testability and reduced work later.

Accessibility and licensing choices also evolved. We initially considered richer audio narration, but shifted to simpler multimodal cues (clear text plus parallel visual/audio signals) to avoid asset/licensing risk and fit the timebox

Resourcing adjustments were needed when availability fluctuated. We concentrated map work with one owner and split scoring/events, while pairing on fragile areas (input handling, collisions) and small refactors (e.g., extracting InputHelper) were scheduled inside tasks to reduce future integration risk without creating separate milestones.

Overall, the plan stayed within the original milestones with two notable adjustments: the event-system skeleton slipped by a few days and was recovered by trimming non-essential polish. Weekly progress on the website should reflect these changes with brief rationales.

EDITS:

MJ_METHODOLOGY

The original team did not specify any software development methodology process.Instead, the methodology method was implicit rather than defined. Knowing that the previous team used GitHub (such as pull request, issues etc..) and had planned dates for when specific areas are finished in the project shows that this team took an informal agile approach and made it unclear on how work was assigned to who.

In Assessment 2, it will be specified on who is responsible for each task. Our team adopts an Agile methodology approach using scrums principles to help guide with the implementation process, software maintenance and change management activities.

The process:

We are using an **Agile Methodology** approach, which emphasises on iterative development where we gain feedback from the customer to respond on what needs changing and improve this to create a new and improved prototype. This approach is suitable for Assessment 2 because this project is inherited from Assessment 1 and requires ongoing maintenance, refinement and continuous integration. In order for the Agile methodology to flourish successfully the team will be using Scrum as the primary framework for organising work, i.e using GitHub issues to set tasks and deadlines for other members of the team, planning iterations and reviewing progress from everyone.

Framework: We will have a master to-do list, to support the backlog. Kiran is responsible for understanding the requirements and prioritising the features to be built. In addition to this, Hannah will manage the Scrum's process by writing up the weekly meeting on what needs to be prioritised next in MEETINGS.md and the outcome.

Product and Sprint Backlog - The product backlog contains all identified tasks by priority. Charles will then translate the product brief into actionable items. There will also be a list of tasks we will be committed to for the current sprint.

Sprint planning - At the start of each week in our meeting we will discuss the highest priority features/tasks from the master to-do list. All members will decide how much work can be completed within the week and break it down into smaller tasks. We will stay in touch over the Instagram group chat of where we are currently at during the week.

Sprint review - At the end of each sprint the members will show what has been completed, whether it was the code or the write up reports. Our goal is to get feedback from the other members to see if there is anything that needs to be improved and to show our progress to the team and customer.

Sprint retrospective - Before starting the next week's sprint, our team will discuss what went well, what didn't and what needs to be changed in the next sprint. Hannah will be responsible for documenting action items the team agrees to implement for the next sprint. This then repeats.

MJ_SYSTEMATIC

Systematic plan:
We're using an extension of their systematic plan but they missed out on adding who was responsible for this task and the priority of this task. To support our weekly sprints for assessment 2 each deliverable/code has new tasks that will be finished weekly building onto the due date. Our team agreed on an internal completion deadline January first (11 days before the actual assessment is due). To allow time for integration, review, testing and final refinements prior to the submission. To allow for easier traceability I've also made sure that each ID links to the name of their report e.g R001 (R standing for change report), I001 for implementenation etc… This is done to make sure when looking at the dependencies column you'll understand how it links to the other deliverables in Assessment 2. The justification of why this is a better approach is in the change report document.

CHANGE REPORT:

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
R101	Complete part(a). Decide on what software maintenance, change management and process that should be used.	10/11/25 - 17/11/25	High - Crucial for deliverables so we could have weekly sprints to complete the required tasks.	Part(a) of Change report completed.	None	Hannah and Kiran
R102	Complete requirements area in part b	17/11/25 - 24/11/25	High - Guides the implementation members on what else they need to add to our game for assessment 2.	New requirements added, new report for Req1_group 7 doc	T011	Kiran
R103	Complete Architecture area in part b	17/11/25 - 24/11/25	High - Likewise with requirements it helps people on implementation on what type of classes needs to be implemented.	New class diagram implemented. New sequence diagram implemented. For Arch1_group 7 doc	T011	Hannah
R104	Complete Method and planning area in part b	10/11/25 - 10/01/26	Low - It is important for tracking progress, but did not need to be properly implemented and refined right away unlike the other areas in the change report as there was a doc as a draft of the progress which could be refined later on into the main document. (Now)	Extension of systematic plan and process defined and implemented. For Plan1_group 7 doc	T011	Hannah
R105	Complete Risk assessment area in part b	17/11/25 - 24/11/25	High - Defined the project and how it should go about avoiding any risks necessary.	New risks added. For Risk1_group 7 doc	T011	Kiran

IMPLEMENTATION

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
I001	Fixing any bugs, Sorting out the tiled map due to it being a bit unorganised.	10/11/25 - 17/11/25	High - This was sorted out to make it so it was easier to implement events using the software Tiled. Bugs were fixed to make sure events and other features will be easy to implement with no worries.	Bugs were fixed, removed all event classes to keep it to one class that handles all of them.	T011, R001	Charles and Ben
I002	Achievements and achievement screen added. A name screen was added for the leaderboard later.	17/11/25 - 24/11/25	High - Part of the requirements that come from the product brief that implementation need to cover.	Achievements and Name screen added.	T011, R101, R102	Charles
I003	A leaderboard screen was added. Added ability to restart and return to the main menu. New positive and negative events were added.	1/12/25 - 8/12/25	High - Players would have to reload the game in order to restart if there wasn't an ability to do so. The leaderboard and events are also part of the product brief.	New leaderboard, restart and return to main menu ability. New speed boost and slowness coffee for positive and negative events.	T011, R101, R102	Charles
I004	Added new hidden event, negative event	8/12/25 - 16/12/25	High - Events are part of the product brief.	New Yeti event thats negative and exam in progress room hidden event	I001, I002, I003, R101	Ben
I005	Added Event tracker, coins, negative event and hidden event	16/12/25 - 23/12/25	High - Events are part of the product brief.	A new troll door negative event that can’t be opened even with a key and finding longboi has been changed to a hidden event. Yeti coins added	I001, I002, I004, R101	Ben

SOFTWARE TESTING

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
S001	Complete part(a) of software testing. Define the test strategy	20/11/25 - 27/11/25	High - Guides the members on the testing deliverable to know what testing method they'll need to use in implementation.	Part a completed Testing method and approach provided	None	Dash
S002	Setting up testing infrastructure for LibGDX	24/11/25	High - prevents LibGDX crashing during testing	Configured test environment	S001	Dash
S003	Adding headless unit tests and asset validation tests	17/11/25 - 23/12/25	High - Verified the main game logic and asset integrity, prevents crashes and any incorrect gameplay behaviour.	JUnit test classes	S001, S002	Dash, AbualHassan
S004	Implement UI/functional tests	17/11/25 - 23/12/25	Medium - Ensures everything in the system works but depends on S001,S002 first.	UI test classes	S001, S002, S003	Dash, AbualHassan
S005	Implement manual tests IF NECESSARY	17/11/25 - 23/12/25	Low - most tests are automated. Only manual tests will be conducted if it can't be auto tested.	Manual test log	S003, S004	Dash, AbualHassan
S006	Produce the execution test report and traceability for requirements.	17/11/25 - 23/12/25	High - shows verification of requirements and evidence of what is correct to submit for Assessment 2	Final testing report. Test2.pdf	S003, S004, S005	Dash, AbualHassan

USER EVALUATION

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
U001	Set up Consent forms, User questions, observation sheet and prompts to evaluate the users.	4/12/25-11/12/25	High - Needed to conduct the user evaluation	Consent forms, questions, observation and prompt sheets are completed.	None	Harley
U002	Conduct the user evaluation with 8 participants.	11/12/25	High - We need to understand what feedback is received so we can make changes to make the quality of the game better during the implementation process.	8 Participants with successful User evaluation for each.	U001	Everyone (each team member conducted the evaluation to one other member in the cohort.
U003	Write up part (a) Including its procedure, justification, recruitment and data collection and tools	4/12/25-11/12/25	High - is needed for deliverables for Assessment 2 and provides documentation of how the user evaluation was conducted.	Part a completed specifying its tools data collection and procedure etc..	U001	Hannah, Harley
U004	Write up part(b) summarising its user evaluation in a table including its usability problem, where it was observed, severity rating and any notes	15/12/25	High - shows evaluation results, displaying usability problems and allows implementation team to respond to feedback	Part b completed, Implementation team used this feedback to fix any bugs or add other features	U002, U003	Hannah, Harley

CONTINUOUS INTEGRATION

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
C001	Set up the initial GitHub Releases, Packr CI pipeline for building and testing	10/11/25	High - Establishes automated building and testing early to detect any integration issues to make sure the code is stable.	New working CI pipeline that builds the project.	GitHub repository, T011	Cassie
C002	Set up cross-platform compilation and testing using the matrix strategy for variables.	16/11/25 - 23/12/25	High - makes sure that games runs properly on Windows. Meets the NFR_SYSTEM_RESTRICTIONS requirement	Automated builds and test results on every push	C001	Cassie
C003	Write up part(a) of CI.	29/12/25	Medium - Required to have a well documented CI design, pipelines, inputs/outputs for Assessment submission for deliverables	Part (a) completed	C001, C002	Cassie
C004	Write up part(b) of CI	31/12/25	Medium - Needed to explain the technical implementation of CI's pipelines and what tools were used for Assessment submission for deliverables	Part (b) completed CI2.pdf made	C001, C002	Cassie

WEBSITE

ID	Task	Date	Priority and why	Output of task	Dependencies	Assigned to:
W001	Download previous group's website and add clear navigation for Assessment 2 deliverables.	10/11/25 - 17/11/25	Medium - could've been done any time	The inherited website has new navigation for Assessment 2 deliverables.	T011	Abualhassan
W002	Add all deliverables into website	06/01/26 - 10/01/26	High - All deliverables, code and executable JAR is needed to be displayed on the website.	All Assessment 2 deliverables, code and executable JAR have been implemented into the website into the correct section.	All dependencies	Abualhassan