Risk Assessment and Mitigation

Cohort 2 Team 1

Ahmet Abdulhamit

Zoey Ahmed

Tomisin Bankole

Alanah Bell

Sasha Heer

Oscar Meadowcroft

Alric Thilak

Risk Management Process:

The team used a lightweight 4-step risk management process as the first version of the project is small scale. This meant a simple, iterative review cycle kept the process efficient without any excessive documentation overhead and ensured that risks were continuously identified, assessed and monitored throughout development.

1. **Identification:** Risks were identified during weekly sprint planning meetings through team brainstorming and reviewing the previous sprint issues discovered.
2. **Analysis:** After being identified, each risk was assessed qualitatively using a 1-5 scale for likelihood (5 = likely) and another 1-5 scale for impact severity (5 being the highest).
3. **Planning/Mitigation:** For each risk the team proposed practical actions to take to reduce the likelihood and impact of every risk identified.
4. **Monitoring:** At the end of each sprint, the risk register was reviewed and the impact/likelihood scores were altered depending on situation changes. Each risk was also assigned an owner responsible for tracking it and updating its status during reviews.

Risk Register Format Justification:
Formatting: The group used a tabular register to manage risks. The tables are grouped by risk type (technical, organisational, scheduling, quality) to help keep each one concise, informative and easy to maintain throughout the project. Each risk entry records what the risk is, likelihood, impact, mitigation and ownership.

Risk Register

Likelihood Key:
Likely = 5/5
Possible = 4/5
Unlikely = 3/5
Rare = 2/5
Very Rare = 1/5

Impact Key:
Very high = 5/5
High = 4/5
Moderate = 3/5
Low = 2/5
Very Low = 1/5

Technical Risk ID:	Risk	Likelihood	Impact and Severity	Mitigation	Ownership
TR1	New tools	Likely	3 - Likely that there will be small bursts of defects and reworks from misconfiguration. New tools could cause early uncertainty and slower delivery.	Coding members begin to practise and ‘play’ with the new tools from week1. This early familiaration will help ease uncertainty before the main coding begins.	The development team learns new tools. The Scrum Master creates practice sessions and watches progress. Method Leads focus is documentation.

TR2	Members do not show up to meetings.	Likely	4 - May delay decisions or cause tasks to become misaligned. This will force reworks and idle ‘blocked’ time. This would delay tasks and testing time. This may lead to schedule slippage as the live feedback loop can't work with poor coordination and missed touchpoints.	Using Whatsapp as a continuous communication channel. Each member should post the progress and any roadblocks. This should also be daily. This will maintain the coordination even if attendance drops. Tasks won't stall.	Scrum Master structures meetings and ensures engagement. Team members update through WhatsApp and Google Docs. Method Leads check sprints and records patterns in attendance.
TR3	Integration issues	Possible	3 - This could trigger rework cycles and delay the next demo. It is likely that integration problems will convert into re-works (extra time and costs) and also schedule slips.	By adopting integration with small and frequent merges to a shared branch where every merge must compile. If we schedule a weekly integration sprint it will help catch incompatibilities and mean fixes are easier.	Developers manage code merging and testing. Architects make sure UML and design align with the codebase. Scrum Master schedules weekly integration checks(sprints) and creates meetings for issue resolution.
TR4	Learning curve for LibGDX and UML tools	Likely	3 - Slower development of features while we learn APIs and modelling notation. A weak UML quality will risk miscommunication - ultimately leading to a rework.	Focus on understanding UML examples online and from lectures. Short internal meetings to discuss and help each other understand will improve the overall team's clarity of UML.	Scrum master will ensure understanding of UML examples by continuously discussing members' work.
TR5	Performance limitations	Possible	3 - This will force late optimisation or feature cuts. This could result in a worse user experience and missed targets.	If we define a performance baseline early on and keep graphics light with incremental testing (performance) after each feature, we should prevent late surprises.	Scrum master reviews work done after weekly sprints. Method Leads should update the plan accordingly. Members should do work agreed at sprints and communicate delays.
TR6	Version control conflicts	Possible	3 - Time will be lost and potentially stall the team near deadlines.	We should only merge features after code review and resolution. This reduces overlapping edits and prevents conflicts from becoming a crisis.	The scrum master is to maintain the communication channel. Architects should be updating developers. Architecture should make sure documents show the

					changes

Organisational and Team Risk ID:	Risk	Likelihood	Impact and Severity	Mitigation	Ownership
OTR1	People are sick	Likely	4 - Tasks could be delayed or reallocated which will lead to a short-term productivity drop.	We should keep tasks small and share documentation so others can help.	Scrum Master will reallocate work. All members will support key tasks.
OTR2	Communication breakdown	Possible	3 - Misunderstanding could lead to duplicated work (or missed work).	Continuous updates through whatsapp and a clear outline of roles in the documents will help prevent this.	Scrum master will update through WhatsApp. Method Leads will maintain the planning document.
OTR3	Uneven workload	Possible	3 - Some members could be overworked and others may end up idle which will reduce quality and morale.	Weekly sprints provide an opportunity to rebalance tasks.	Scrum Master will rebalance the sprint tasks. Method Leads will track all of the changes and each member should assist as needed.
OTR4	Limited availability	Likely	4 - Less available members will cause sprint goals to not be met.	Continuous communication on whatsapp means we can plan ahead, prioritising core features.	Scrum Master manages scheduling. Team members make absences known early.

Scheduling and Delivery Risk ID:	Risk	Likelihood	Impact and Severity	Mitigation	Ownership
SDR1	Underestimating time needed to complete task	Possible	4 - Could delay the delivery of sprints. Could lead to features being rushed or even dropped.	It would be a good idea to break tasks into smaller substacks. We could also include buffer times in sprint planning.	Scrum Master will lead the planning of sprints. Method Leads will record any issues with estimation. Developers will provide timing input.

SDR2	Missing sprint goals	Possible	3 - This would build pressure and also reduce the visibility of progress.	We should prioritise the minimum viable product tasks first.	Scrum Master monitors the progress of sprints. Method Leads direct the team in prioritising tasks.

Quality and Requirement Risk ID:	Risk	Likelihood	Impact and Severity	Mitigation	Ownership
QRR1	Misinterpretation of customer requirements	Possible	4 - Features will not meet their expectations. This would also waste the developing effort.	We should confirm requirements early with team reviews and ensure clear meeting notes.	Zoey, Sasha and Ahmet should confirm the requirements. Scrum Master should review the requirements.
QRR2	Lack of traceability	Possible	3 - This would make it hard to verify tasks which fit the requirements.	Git commits and continuous updates will be crucial.	Developers will ensure documentation. Developers keep the commits linked. Scrum master checks progress.
QRR3	Lack of testing	Possible	4 - Could lead to undetected bugs or unstable features. This will reduce the quality of features.	We should implement unit tests and run playtests after each sprint.	Developers handle the testing. Scrum Master monitors the inclusion of tests.