Telemetry Game Report

COM2020 - Project 7

Harry Taylor, Luca Pacitti, Emre Acarsoy, Tom Croft, Luca Croci, Will Finney, Kazybek Khairulla

17/02/2026

Contents

1 Executive Summary

TODO Luca C

2 Problem Framing

The monitoring and maintenance of a game is essential for ensuring that players have an enjoyable experience, and that their engagement remains intact. However, game designers are often left to rely on their own intuition and small playtests to achieve this. Relying on subjective assessment makes it difficult for them to identify problem areas of a game where a player's engagement depreciates - such as difficulty spikes or features that provoke inequity between play styles.

This project integrates an indie game with a telemetry application, generating intuitive telemetry events for several actions which are processed and presented in several ways. The objective is to aid the game designer's understanding of a player's experience, and how they could maintain the game to maximise the overall player experience.

3 Project Backlog

The project was split into three main sections representing the three main areas of work: the documentation, the java application, and the python application. Each of those areas is then further divided into epics and user stories. The epics and stories in this section include both those relevant to sprint 1 and what is planned for sprint 2. Due to the nature of scrum, the epics for sprint 2 may have adapted by the end of that sprint, and so may differ from the epics and stories in this prototype report.

3.1 Documentation

3.1.1 Prototype Report

3.1.1.1 Executive Summary

As a product owner I need an executive summary to give me an insight into what is to come in the report and provide the most important details to me.

3.1.1.2 Problem Framing

As a product owner I need an in depth exploration of the problem so that I am informed of what problem the system should solve and ensure the development team fully understand the problem.

3.1.1.3 Project Backlog

As a scrum master I need a project backlog so I can determine what work is to be done in each sprint.

3.1.1.4 Sprint One Prioritised Requirements

As a product owner I need to know what requirements are being prioritised in sprint one to know that they align with my priorities.

3.1.1.5 Architecture Schema

As a developer I need to know what the project schema is so I can produce code that integrates with it correctly.

3.1.1.6 Data Flow Schema

As a developer I need to know what the flow of data is so I can work with the data in a manner that integrates well with the rest of the system.

3.1.1.7 Telemetry Schema

As a developer I need to know the telemetry schema so I can interact with telemetry events correctly in a way which integrates well with the rest of the system.

3.1.1.8 Initial Evaluation

As a product owner I need an initial evaluation of the prototype so I can see the developers are aware of their product and know what areas to work on in the future to improve it.

3.1.1.9 Sprint Two Prioritised Requirements

As a product owner I need to see what requirements are being prioritised and planned for in sprint two so I can know they align with my priorities.

3.1.1.10 Management

As a product owner I need to see how the team has worked together and what roles each person took on during development so I can evaluate each members performance and ensure the team is working well.

3.1.2 Meeting Minutes

3.1.2.1 Meeting Attendance

As a scrum master I need to know who is attending meetings so I can follow up people not there to ensure they are kept up to date.

3.1.2.2 Work Completed

As a scrum master I need to know what work has been completed in each weekly scrum so I can determine what work needs to be prioritised for the next scrum, and plan for the next sprint.

3.1.2.3 Topics Discussed

As a scrum master I need to know what topics were discussed in the weekly scrum so I can keep developers who could not make it to the meeting updated.

3.1.2.4 Work to be Done

As a scrum master I need to know what work has been assigned to each member so that I can check they're completing work on time, and follow up with them if issues arise.

3.1.3 Risk Register

3.1.3.1 List of Risks

As a stakeholder I need to know what risks there are in the software so I can make informed decisions about the adoption and use of the software.

3.1.3.2 Risk Mitigations

As a stakeholder I need each risk to have a mitigation plan and owner so that risks are actively managed and their impact reduced.

3.1.4 Ethical and Legal Considerations

3.1.4.1 Privacy and Data Protection Analysis

As a product owner I need the software to handle data in full legal compliance and respect the privacy of users to foster user trust, maintain regulatory compliance and minimise privacy risk.

3.1.4.2 Consent and Disclosure Analysis

As a system user I need the software to obtain my consent for the collection of data, and disclose to me what data is collected and why so that I can make informed decisions about my data.

3.1.4.3 Accessabilty Analysis

As a system user with accessability requirements I need to the software to be accessible to me by complying with modern accessability standards so that I can use it to best advantage.

3.1.4.4 Intelectual Property and Licensing Implications

As a product owner I need to know what the licensing and intellectual property constraints of the software are so that I can use it in full legal compliance and honor the rights of the intellectual property owners.

3.1.5 Project License

3.1.5.1 Determine the Project License

As a product owner I need to know what license the product falls under so I that its usage complies with the license.

3.1.6 Software and Data Inventory

3.1.6.1 Software Inventory

As a maintainer I need to know what software is a direct dependency of the system as well as this software's license, provenance, cost model and the version in use, so that I can build, test, extend and maintain the system.

3.1.6.2 Data Inventory

As a maintainer I need to know what data the software directly depends on as well as this data's license, provenance, cost model and the version used, so I can build, test, extend and maintain the system.

3.1.7 Deployment Guide

3.1.7.1 Java Application Deployment Instructions

As a client I need instructions on how to deploy and run the Java application so I can perform actions using it.

3.1.7.2 Python Application Deployment Instructions

As a client I need instructions on how to deploy and run the Python application so I can perform actions using it.

3.1.7.3 Automated Test Running Instructions

As a maintainer I need instructions on how to run the automated test suite so that I can test any changes or extensions to the software to ensure their correctness.

3.1.8 Test Evidence

3.1.8.1 Automated Test Evidence

As a product owner I need evidence of the system passing automated tests so that I know the software runs correctly in the tested areas.

3.1.8.2 End to End Test Evidence

As a product owner I need evidence of the system passing end to end tests so that I know the software runs correctly in the tested areas.

3.1.9 Presentation

3.1.9.1 The Problem

As a product owner I need to see that the group are aware of the problem the system will solve so that we both reach a point of understanding and agreement in relation to the problem.

3.1.9.2 Approach

As a product owner I need to see what approach the group took to solving the problem and how they designed their solution so that I can make an informed decision about it's adoption, utility, and effectiveness.

3.1.9.3 Implementation

As a product owner I need to see what path was taken to implement the product and how it was implemented so that I can understand the implementation and see why they made different decisions in relation to the implementation.

3.1.9.4 Evaluation

As a product owner I need to see an evaluation of the prototype so that I am informed of its advantages and limitations and thus can know its utility and where best to use it.

3.1.9.5 Demonstration

As a product owner I need to see a demonstration of the prototype so that I can see that it works, how to use it, and what it offers as well as to evidence what has been done in the project.

3.1.9.6 Limitations and Next Steps

As a product owner I need to see where the project is going to next so that I can see it will be improved in areas of limitation and that the group is on schedule to deliver the complete system.

3.1.10 Operation Guide

3.1.10.1 Operation Instructions

As a client I need an operation guide explaining how to operate the system so I can use it effectively.

3.1.10.2 Maintenance Instructions

As a maintainer I need clear maintenance instructions so I can ensure the game, settings, viewer and telemetry are functioning properly.

3.1.10.3 Troubleshooting Instructions

As a general user I need clear troubleshooting instructions based on clear and understandable error messages so that I can quickly troubleshoot errors and continue playing the game.

3.1.10.4 Extension Guide

As a maintainer I need to know what areas of the system should be the targets of extension and how to extend of the current system so that when the system needs extension those are done in the easiest, least risky way.

3.1.11 Data Management Guide

3.1.11.1 Data Stored

As a data protection compliance officer I need to know what data is stored by the system so I can ensure this data is minimised and in full compliance with the law.

3.1.11.2 Data Format

As a maintainer I need to know what format the data is stored so I can work with the stored data and extend what data is stored.

3.1.12 Scrum Board

3.1.12.1 Backlog, In Progress and Done Sections

As a scrum master I need to know what tasks are to be done, in progress, and completed so I can plan for the next daily scrum.

3.2 Python Application

3.2.1 User Interface

3.2.1.1 Login

As a designer I need a login menu so that I can log into the python application and use it.

3.2.1.2 Dashboards

As a designer I need a dashboard menu so that I can view the various dashboard views of the captured telemetry data.

3.2.1.3 Suggestions

As a designer I need a suggestions menu so that I can view the rule based suggestions produced by the application.

3.2.1.4 Decision Log

As a designer I need to a menu to view the decision log so I can see what design decisions have been made.

3.2.2 Authentication and Acess Control

3.2.2.1 Login

As a general user I need to be able to log in with the same credentials as for the java application so that I can be granted access to the telemetry application in line with my role.

3.2.2.2 Password Reset

As a general user I need to be able to reset my login credentials so that I can recover my login if I lose them.

3.2.2.3 Player Permissions

As a player I need to not be able to access the telemetry application so that I do not view collected telemetry in a way that invades the privacy of others.

3.2.2.4 Designer Permissions

As a designer I need to be able to access the telemetry application so that I can view the telemetry dashboards, design log and rule based suggestions.

3.2.2.5 Developer Permissions

As a developer I need to be able to do anything a designer can do so that I can help develop the game based on information in the telemetry application.

3.2.3 Dashboard Views

3.2.3.1 Simulation View

As a designer I need to see the simulation predictions to the effects of my design decisions as dashboard views so that I can make informed design decision.

3.2.3.2 Funnel View

As a designer I need to be able to view a stage by stage completion funnel so that I can see at what stages player runs drop off and what the failure rate is of each stage.

3.2.3.3 Difficulty Spikes

As a designer I need to be able to view which stages have unusually high failure rates or take a long time to complete so that I can focus my attention on balancing them.

3.2.3.4 Progress Curves

As a designer I need to view the time to complete each stage and how many coins are accumulated on each stage so that I can better balance these factors.

3.2.3.5 Fairness Indicators

As a designer I need to view a comparison between different play styles on the same stages so that I can determine if the game unfairly rewards one play style over another.

3.2.3.6 Comparison Mode

As a designer I need to view a comparison on key metrics between the different difficulty modes so that I can ensure the jump in difficulty between modes is appropriate.

3.2.4 Design Suggestions

3.2.4.1 Rule Based Design Suggestions

As a designer I need the telemetry application to provide rule based design suggestions to help me improve the game's balance.

3.2.5 Telemetry Events

3.2.5.1 Read Telemetry Events

As a designer I need the python application to be able to read the same telemetry event set that the java application produces so that I can view them in the application.

3.2.5.2 Validate Telemetry Events

As a designer I need the python application to be able to validate telemetry events so it can handle anomalous telemetry events gracefully.

3.2.6 Seeded Dataset

3.2.6.1 Telemetry Events

As a developer I need a large number of seeded telemetry events so that I can use them to test the system handles telemetry in the expected manner.

3.2.6.2 Stage Variety

As a developer I need the events in the dataset to come from all stages in the game so that I can test the systems can handle events from all stages.

3.2.6.3 Difficulty Variety

As a developer I need the events in the dataset to come from all difficulty configurations so that I can test the systems can handle events from all difficulties.

3.2.6.4 User Variety

As a developer I need the events in the dataset to come from a variety of user IDs so that I can test the system can handle events form a variety of user IDs.

3.2.6.5 Session Variet

As a developer I need the events in the dataset to come from a variety of different sessions so that I can test the system can handle events from a variety of sessions.

3.2.6.6 Anomalous Telemetry Events

As a developer I need some of the seeded telemetry events to contain anomalous data so that I can test the system is able to gracefully handle this data.

3.2.6.7 Balancing Decisions

As a developer I need a set of sample balancing decisions within a sample decision log, that are a combination of good and bad decisions so that I can test the system is able to process the decision log.

3.3 Java Application

3.3.1 User Interface

3.3.1.1 Login

As a general user I need a login screen so I can log into the java game to use it.

3.3.1.2 Main Menu

As a general user I need a main menu so that I can have a central access point to starting runs, viewing my progress and accessing my settings.

3.3.1.3 Settings

As an authenticated user I need a settings menu to allow me to change settings, design parameters, and assign roles based on what role I have.

3.3.1.4 Start Run

As a general user I need a menu to allow me to select which difficulty for a game run I want to start.

3.3.1.5 Encounter

As a general user I need an encounter menu so that I can select which attack abilities to use against enemies, view enemy health and view my health, lives, magic and coins.

3.3.1.6 Shop

As a general user I need a shop menu so that I can view which upgrades are available for me to purchase, how much they cost, and how many coins I have to buy them with.

3.3.1.7 End Run

As a general user I need an end run menu so that I can view which stage I reached, how many times I died, how many coins I finished with, and how long the run was.

3.3.2 Autheentication and Acesss Control

3.3.2.1 subsubsection

Login As a general user I need to be able to login with a username and password so that the telemetry events produced by me are associated with my user ID.

3.3.2.2

Login Reset As a general user I need to be able to reset my login so if I forget my login credentials I can recover my account by resetting them.

3.3.2.3 Player Permissions

As a player I need to be able to login, change whether telemetry is enabled, view my run progress, view a telemetry disclosure, play any number of game runs so that I can play the game and consent to the collection of my telemetry data.

3.3.2.4 Designer Permissions

As a designer I need to be able to do anything a player can do, as well as view and edit design parameters in the settings screen so that I can edit the game's design and balance.

3.3.2.5 Develoer Permissions

As a developer I need to be able to do anything a designer can do, as well as assign roles to different users so users can gain roles other than the player role.

3.3.3 Settings

3.3.3.1 Telemetry

As a general user I need a setting that allows me to enable and disable telemetry that is enabled by default so I can withdraw my consent for the collection of telemetry events relating to my play.

3.3.3.2 Design Parameters

As a designer I need to view and set the values of the design parameters of the system so that I can balance the game.

3.3.3.3 Simulation Mode

As a designer I need to view a simulation of the game so that I can see what the effect of changing the design parameters are on the game's balance.

3.3.3.4 Role Asignment

As a developer I need to be able assign roles to different users so that designers are given the designer role and developers are given the developer role, giving them the correct authorisation.

3.3.4 Gameplay Loop

3.3.4.1 Start Run

As a general user I need to be able to start a run so that I can play the game.

3.3.4.2 Choose Difficulty

As a general user I need to be able to choose the difficulty for my run so that it is an appropriate difficulty for my skill level.

3.3.4.3 Be in an Encounter

As a general user I need to be able to enter an encounter after starting a run, and go though one encounter per stage so that I am challenged by the game.

3.3.4.4 Use Physical Attack Abilities

As a general user in an encounter I need to be able to select any physical attack ability I have and use them against any enemy in the encounter so that I can make interesting decisions in the game that affect how successful I am.

3.3.4.5 Use Magical Attack Abilities

As a general user in an encounter I need to be able to select any magic attack ability I have, view it's magic cost and if I have enough magic use it against any enemy in the encounter so that I can make interesting decisions in the game that affect how successful I am.

3.3.4.6 Take Turns

As a general user in an encounter I need to be able to take my turn before all enemies take their turn, then have each enemy in the encounter take their turn so that the encounter is fair.

3.3.4.7 Kill Enemies

As a general user in an encounter I need attack abilities to reduce enemy health and when their health is reduced to 0 or less they die and can no longer take turns so that I can prioritise dangerous enemies and succeed in encounters.

3.3.4.8 Complete an Encounter

As a general user in an encounter I need to complete the encounter I'm in by killing all the enemies so that I can progress to the next stage.

3.3.4.9 View Shop

As a general user after completing an encounter that was not the final encounter, I need to view a number of upgrades from the list of all upgrades that I have not yet purchased, that are available for purchase in the shop and their cost so that I can buy upgrades and empower my character.

3.3.4.10 Purchase Upgrades in the Shop

As a general user in the shop I need to be able to purchase upgrades that I have enough coins to buy and obtain the passive or attack abilities associated with the upgrades so that I can customise my character.

3.3.4.11 Complete a Run

As a general user after I complete the final encounter, run out of lives, or quite a run, I need the run to be completed so that I can view the run statistics and go back to the main menu.

3.3.5 Game Mechanics

3.3.5.1 Coins

As a general user I need to collect coins at the end of each encounter so that I can spend them in the shop.

3.3.5.2 Health

As a general user I need my character and enemies to have health that is reduced when they take damage so that my character and enemies can die.

3.3.5.3 Lives

As a general user I need my character to have lives so that if they die the run doesn't end until they run out of lives.

3.3.5.4 Damage Types

As a general user I need several damage types so that different abilities of that damage type can affect different entities differently and the amount of damage of each type they deal can be upgraded.

3.3.5.5 Magic

As a general user in an encounter I need to be able to gain magic at a predefined rate at the start of each of my turns so that I can use magic abilities.

3.3.5.6 Passive Abilities

As a general user I need to be able to obtain passive abilities from the shop that provide passive effects such as increasing my character's damage of a specific damage type or reducing the damage they take of a specific type so that I can upgrade my character.

3.3.5.7 Attack Abilities

As a general user I need to be able to obtain attack abilities from the shop that can be used in the encounter, targeting enemy entities and dealing damage so that I can upgrade their attacks and kill stronger enemies.

3.3.5.8 Difficulties

As a general user I need runs to be able to choose if my run is easy, moderate, or hard difficulty, where each difficulty has a different value for each design parameter so that I can play a run that is of an appropriate difficulty for my skill level.

3.3.6 Telemetry Events

3.3.6.1 Event Writing

As a designer I require the Java application to write telemetry events to the telemetry event store if the user has telemetry enabled, so I can view them in the python application.

3.3.6.2 Event Validation

As a designer I require the Java application to validate the events it writes to the event store so that the number of anomalous events written is minimised.

3.3.6.3 Start Session

As a developer I need to view information relating to the start of a session (run) so that I can know what difficulty the session is.

3.3.6.4 Normal Encounter Start

As a designer I need to view information relating to the start of a normal encounter so that I can know what encounter is taking place, on which stage and on what difficulty.

3.3.6.5 Normal Encounter Fail

As a designer I need to view information relating to the failure of a normal encounter so that I can know what encounter was failed, what difficulty it was on, what stage the encounter was on, and how many lives the player has left on the run.

3.3.6.6 Normal Encounter Complete

As a designer I need to view information relating to the completion of a normal encounter so that I can know what encounter was completed, what difficulty the encounter was, what stage it was on, and how much health the player finished with.

3.3.6.7 Boss Encounter Start

As a designer I need to view information relating to the start of a boss encounter so I can know what encounter is taking place, on which stage and on what difficulty.

3.3.6.8 Boss Encounter Fail

As a designer I need to view information relating to the failure of a boss encounter so that I can know what encounter was failed, what difficulty it was on, what stage the encounter was on, and how many lives the player has left on the run.

3.3.6.9 Boss Encounter Complete

As a designer I need to view information relating to the completion of a boss encounter so that I can know what encounter was completed, what difficulty the encounter was, what stage it was on, and how much health the player finished with.

3.3.6.10 Gain Coin

As a designer I need to view information relating to the player gaining coins so that I know how many coins were gained, on what difficulty they were gained on, what encounter provided them and what stage they were gained on.

3.3.6.11 Buy Upgrade

As a designer I need to view information relating to the player purchasing an upgrade from the shop so that I know what stage the shop is associated with, what upgrade was bought and how many coins it cost.

3.3.6.12 End Session

As a designer I need to view information relating to the end of a session (run) so that I know how long the session was.

3.3.6.13 Settings Change

As a designer I need to view information relating to changing a setting value so that I know what setting was changed and to what value.

3.3.6.14 Kill Enemy

As a designer I need to view information relating to the death of an enemy so that I know what encounter the enemy was a part of, what difficulty the run was in, what stage the encounter was for, and what enemy was killed by the player.

4 Sprint 1 Prioritised Requirements

4.1 Documentation

4.1.1 Prototype Report

4.1.1.1 Executive Summary

Value: 2 **Priority:** Must
Acceptance Criteria

- The summary appears at the start of the report.
- The summary clearly states what is delivered in the prototype.
- The summary clearly states why prototype features exist in the prototype.
- The summary clearly states the most important risks of the prototype.
- The summary clearly states the mitigations for those risks.

4.1.1.2 Problem Framing

Value: 3 **Priority:** Must
Acceptance Criteria

- The problem framing clearly explains the problem.
- The problem framing clearly states why the problem is important.
- The problem framing clearly states what the goal of the project is.
- The problem framing must state who is affected by the problem.

4.1.1.3 Project Backlog

Value: 3 **Priority:** Must
Acceptance Criteria

- The project backlog must clearly state which epics are required to complete the project.
- The project backlog must clearly state which user stories are associated with each epic.

4.1.1.4 Sprint One Prioritised Requirements

Value: 3 **Priority:** Must
Acceptance Criteria

- Each epic in the project backlog must be stated in the requirements.
- Each story within each epic must be stated in the requirements.
- Each story should have a value associated with it, based on how much work it is to complete the story, the significance of the risks associated with it, the technical changes required to complete it, how much depends on it, how complex it is to complete, and the value it's completion adds to the project.

- Each story must have an acceptance criteria associated with it, which when all criteria are met the story is considered complete for sprint 1. Criteria may change between sprints as sprint requirements change.
- Each story must have a priority associated with it for that sprint. The priority should either be Must, Should, Could or Won't.
- Each story must be able to link to a prototype feature and test.

4.1.1.5 Architecture Schema

Value: 3 **Priority:** Should
Acceptance Criteria

- Each class within the Java application must have it's purpose and use case explained.
- There must be a UML diagram of the Java application showing how classes interact.
- Each component of the Python application must have it's purpose and use case explained.
- All design choices must be justified and explained.
- The limitations of all design choices must be stated, and why they're acceptable must be explained.

4.1.1.6 Data Flow Schema

Value: 2 **Priority:** Should
Acceptance Criteria

- Each component that accesses data should be explained, describing what data it accesses, what it does with it, and why it needs to access this data.
- Each data store should be explained, stating what data it stores and in what format.

4.1.1.7 Telemetry Schema

Value: 3 **Priority:** Must
Acceptance Criteria

- Each telemetry event must be described by the schema.
- Each telemetry event must have all it's fields described by the schema.
- Each field must have its domain described by the schema. For enumerated fields all valid values must be described.

4.1.1.8 Initial Evaluation

Value: 5 **Priority:** Must
Acceptance Criteria

- Several metrics and measures of success should be evaluated. Where applicable they should be compared with the baselines of other solutions to the problem.

- The method coverage of the test suite should be evaluated to determine how thoroughly the system is examined, and thus how confident we can be that defects will be discovered early.
- The time between the user inputting what attack ability they use, and the command line outputting the result of their attack and the enemies' turn to measure the responsiveness of the game.
- The memory usage of the prototype must be less than 1GB, and ideally as small as possible, to ensure it can run on devices with small amounts of memory.
- The ability for the prototype to handle invalid input gracefully as to make it accessible and secure.
- The ability for the prototype to handle exceptions and edge cases gracefully to ensure the user experience is smooth and error free.
- The ability for the prototype to handle anomalous telemetry events gracefully to ensure the designer experience is smooth and error free.
- The usability of the prototype to allow designers to locate issues with the game and make informed changes within a short timeframe to ensure it is a useful tool for designers.
- The effectiveness of the rule based suggestion to ensure they provide help to designers.
- The limitations of the evaluation, including biases in the methodology, and whether the tests and evaluation methods generalise such that they're comparable to real world scenarios.

4.1.1.9 Sprint Two Prioritised Requirements

Value: 3 **Priority:** Should
Acceptance Criteria

- Each epic in the project backlog must be stated in the requirements.
- Each story within each epic must be stated in the requirements.
- Each story should have a value associated with it, based on how much work it is to complete the story, the significance of the risks associated with it, the technical changes required to complete it, how much depends on it, how complex it is to complete, and the value it's completion adds to the project.
- Each story must have an acceptance criteria associated with it, which when all criteria are met the story is considered complete for sprint 2. Criteria may change between sprints as sprint requirements change.
- Each story must have a priority associated with it for that sprint. The priority should either be Must, Should, Could or Won't.
- Each story must be able to link to a final product feature and test.

4.1.1.10 Management

Value: 2 **Priority:** Should
Acceptance Criteria

- The management section should outline what roles each member had.
- It should outline what tasks members owned, and what tasks they contributed to.
- It should outline in what areas the group worked well together.
- It should outline what challenges the group faced working together.
- It should outline how those challenges were solved.

4.1.2 Meeting Minutes

4.1.2.1 Meeting Attendance

Value: 1 **Priority:** Must
Acceptance Criteria

- For each meeting, the group members who did attend the meeting are noted.
- For each meeting, the group members who did not attend the meeting are noted.

4.1.2.2 Work Completed

Value: 1 **Priority:** Must
Acceptance Criteria

- For each meeting, if a group member was present the work they completed during the previous weekly scrum should be outlined.

4.1.2.3 Topics Discussed

Value: 1 **Priority:** Must
Acceptance Criteria

- For each meeting, what topics were presented by group members and discussed by the group must be recorded.

4.1.2.4 Work to Be Completed

Value: 1 **Priority:** Must
Acceptance Criteria

- For each meeting, the work assigned to each group member (whether they were there or not) must be recorded.

4.1.3 Risk Register

4.1.3.1 List of Risks

Value: 2 **Priority:** Must
Acceptance Criteria

- A list of all risks associated with role based access control should be included.
- A list of all risks associated with authentication should be included.
- A list of all risks associated with the storage of telemetry data should be included.
- A list of all risks associated with design parameters should be included.

- A list of all risks associated with logging should be included.
- A list of all risks associated with the storage of user information should be included.

4.1.3.2 Risk Mitigations

Value: 2 **Priority:** Must
Acceptance Criteria

- For each risk, the mitigation strategy in use should be described and explained.

4.1.4 Ethical and Legal Considerations

4.1.4.1 Privacy and Data Protection Analysis

Value: 1 **Priority:** Must
Acceptance Criteria

- Must discuss what data is stored about users.
- Must discuss how the data storage complies with law.
- Must discuss how collected data is pseudonymised.
- Must discuss how access to this data is limited to only those it's relevant to.

4.1.4.2 Consent and Disclosure Analysis

Value: 1 **Priority:** Must
Acceptance Criteria

- Must discuss what the data is used for.
- Must discuss how consent is handled.
- Must discuss how this consent and disclosure policy complies with law.
- Must discuss how this policy affects the user experience.

4.1.4.3 Accessabilty Analysis

Value: 1 **Priority:** Should
Acceptance Criteria

- Must discuss the importance of accessability.
- Must discuss what systems are in place to support the accessability of the software.
- Must discuss what areas the system can improve upon for sprint 2.

4.1.4.4 Intelectual Property and Licensing Implications

Value: 1 **Priority:** Must
Acceptance Criteria

- Must discuss the intellectual property rights of the developers.
- Must discuss the intellectual property rights of the product owner.
- Must discuss the licensing considerations for the software and data the system depends on.
- Must discuss the intellectual property rights of sources the systems draws inspiration from or is similar to.

4.1.5 Project License

4.1.5.1 Determine the Project License

Value: 1 **Priority:** Must
Acceptance Criteria

- A license for the project must be determined.
- The license must comply with the licensing requirements of all dependencies.

4.1.6 Software and Data Inventory

4.1.6.1 Software Inventory

Value: 2 **Priority:** Should
Acceptance Criteria

- An inventory of all software components the system directly depends on must be produced.
- This must contain the license of each dependency.
- This must contain the cost model of each dependency.
- This must contain the provenance of each dependency.
- This must contain the version of each dependency the system uses.

4.1.6.2 Data Inventory

Value: 1 **Priority:** Should
Acceptance Criteria

- An inventory of all data the system directly depends on or uses must be produced.
- This must contain the license of each dependency.
- This must contain the cost model of each dependency.
- This must contain the provenance of each dependency.
- This must contain the version of each dependency the system uses.

4.1.7 Deployment Guide

4.1.7.1 Java Application Deployment Instructions

Value: 1 **Priority:** Must
Acceptance Criteria

- The guide must contain clear instructions for how to setup and run the Java application.
- There may be instructions for different systems, but there should be a way to run the application on any desktop system that runs Windows, MacOS or Linux.

4.1.7.2 Python Application Deployment Instructions

Value: 1 **Priority:** Must
Acceptance Criteria

- The guide must contain clear instructions for how to setup and run the Python application.
- There may be instructions for different systems, but there should be a way to run the application on any desktop system that runs Windows, MacOS or Linux.

4.1.7.3 Automated Test Running Instructions

Value: 1 **Priority:** Must
Acceptance Criteria

- There should be clear instructions on how to run the test suite on the source code.
- The version of the testing framework should be provided.
- The running instructions should allow the running of the test on any desktop system that runs either Windows, MacOS or Linux.

4.1.8 Test Evidence

4.1.8.1 Automated Test Evidence

Value: 2 **Priority:** Must
Acceptance Criteria

- There must be 5 automated tests.
- For each automated test it should be outlined what component is being tested.
- For each automated test it should be outlined what the expected output is.
- For each automated test the actual output of the system should be provided, and thus evidence that the system passes the test.
- The system must pass all automated tests.

4.1.8.2 End to End Test Evidence

Value: 1 **Priority:** Must
Acceptance Criteria

- There must be one end to end test.
- Evidence that the test is successful should be given.
- The happy path and failure cases for the the test must be described.
- The expected results of the test should be described.

4.1.9 Presentation

4.1.9.1 The Problem

Value: 2 **Priority:** Must
Acceptance Criteria

- Should outline what the problem is.
- Should outline why the problem is relevant.
- Should outlie the scope of the problem.
- Should outline the scope of the solution.
- Should outline what assumptions were made about the problem.
- Should outline who the intended users of the solution are.

4.1.9.2 Approach

Value: 2 **Priority:** Must
Acceptance Criteria

- Should outline the architecture of the solution.
- Should speak about similar solutions.
- Should outline measures used to evaluate the system.

4.1.9.3 Implementation

Value: 2 **Priority:** Must
Acceptance Criteria

- Should outline the key components of the solution.
- Should outline the key features of the implementation.
- Should outline why key design decisions were made.

4.1.9.4 Evaluation

Value: 2 **Priority:** Must
Acceptance Criteria

- Should outline what issues were faced during implementation.
- Should outline how well the system performs against measures outlined in approach, and compare it to existing solutions.
- Should provide clear evidence of how well the prototype works.

4.1.9.5 Demonstration

Value: 3 **Priority:** Must
Acceptance Criteria

- Should take around 3 minutes to complete.
- Should demonstrate the core functionality of the system.
- Should be rehearsed before the live demo, to ensure it works correctly.
- Should have a backup demonstration/video if the demo goes wrong.

4.1.9.6 Limitations and Next Steps

Value: 2 **Priority:** Must
Acceptance Criteria

- Should outline the areas in which the prototype is weakest.
- Should briefly outline what is to be done in the next sprint.

4.1.10 Operation Guide

4.1.10.1 Operation Instructions

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.10.2 Maintenance Instructions

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.10.3 Troubleshooting Instructions

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.10.4 Extension Guide

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.11 Data Management Guide

4.1.11.1 Data Stored

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.11.2 Data Format

Value: 0 **Priority:** Won't
Acceptance Criteria N/A

4.1.12 Scrum Board

4.1.12.1 Backlog, In Progress and Done Sections

Value: 3 **Priority:** Should
Acceptance Criteria

- There must be a card for each story in the project backlog defined in this report.
- There must be a card for each group of acceptance criteria (task) that varies between sprints.
- There must be epics created on the board for each epic defined in the project backlog in this report.
- There must be a backlog section for tasks that have yet to have been started.
- There must be an in progress section for tasks being worked on. Each task in this section must be assigned to at least one group member.
- THere must be a done section for tasks that are complete. Each task in this section must be assigned to at least one group member.

4.2 Python Application

4.2.1 User Interface

4.2.1.1 Login

4.2.1.2 Dashboards

4.2.1.3 Suggestions

4.2.1.4 Decision Log

4.2.2 Authentication and Acess Control

4.2.2.1 Login

4.2.2.2 Password Reset

4.2.2.3 Player Permissions

4.2.2.4 Designer Permissions

4.2.2.5 Developer Permissions

4.2.3 Dashboard Views

4.2.3.1 Simulation View

4.2.3.2 Funnel View

4.2.3.3 Difficulty Spikes

4.2.3.4 Progress Curves

4.2.3.5 Fairness Indicators

4.2.3.6 Comparison Mode

4.2.4 Design Suggestions

4.2.4.1 Rule Based Design Suggestions

4.2.5 Telemetry Events

4.2.5.1 Read Telemetry Events

5 Architecture Schema

5.1 Overview

The project is split into two applications, one written in Python and the other written in Java. The reason for this is we determined that much of the functionality of displaying graphs of data, and handling data in general was much better suited to python than to any other programming language we were confident with. Furthermore we determined that java would be much better suited to create the video game in than python given the more rigorous syntax and typing, and better object oriented support. From there we divided the responsibility of the system between the two applications as described below.

Python

- **Dashboard views**, as the handling of data and generation of graphs is very well supported in Python.
- **Design suggestions**, as the Python application already has to read the telemetry data, and thus components processing it are more simple to implement in the Python application.
- **Simulation views**, as the simulation generates telemetry identical to that of users, and as the Python application can already handle user made telemetry and display it, the functionality for displaying simulation generated telemetry is almost entirety already in the Python application, requiring minimal additional work to implement.
- **Decision Log View**, although not to be implemented in sprint one, it is planned that the Python application will be the access point to the design log, as this allows the cross-examination of logged decisions and the telemetry data that is used to justify those decisions.

Java

- **Role Assignment**, as Java is a more rigorous language than python, and the application requires more complex role based access control due to the implementation of design parameters, it was determined that the Java application should manage role assignment. Specifically, as it was required for the system to operate, but not specified which role should have this permission in the specification, we chose to give only developers the ability to assign roles as this best fit with their role of maintaining the application.
- **Design Parameters**, as the Java application is the only application influenced by the value of the design parameters, we determined those should be set in the Java application, as well as read and written to a configuration file by the Java application. In the next sprint we plan to add a section to the design parameter setting menu that requires designers and developers to justify their changes, which will be stored in the decision log.
- **Telemetry Setting**, as the java application collects telemetry, it was decided the user setting to disable this would be located in the settings menu of the java application.
- **Video Game**, due to java's extensive support of object oriented programming, and due to the fact that video games are generally very well suited to this paradigm it was decided the video game that produces the telemetry events would be coded on the Java application.

5.2 Python Application

TODO harry or emre

5.3 Java Application

5.3.1 Overview

The Java application was first split into front end and back end. From there the responsibilities of the backend were further divided into several singletons the front end would access. Each singleton was given responsibility for a particular area, and that area was further divided between different objects it composited. The front end then uses these singletons to perform the calculations required to run the game, and displays this information to the user.

5.3.2 Design Principles

5.3.2.1 Testability

The Java application was designed with testing as a priority, specifically to allow the extensive use of mock objects. The first way this is achieved is through the use of a modified singleton pattern, where the concrete singleton is nested within a different class that provides access. This allows the concrete singleton to implement an interface, and thus allows the rest of the system to work around that interface rather than the singleton itself. This allows the switching out of the concrete singleton instance for a mock object for testing. Furthermore many of the objects within the java application are built around interfaces rather than implementations, to allow those objects to be swapped out for mock objects for testing.

Another way in which the application is designed with testing in mind is by allowing the directories of all file writes to be modified, to allow a temporary directory to be setup for testing where mock events and configuration files are written and read to, and the resetting the destination of the systems that write files after the test is complete.

5.3.2.2 Extensability

Its already been mentioned about the principle of programming to an interface for the purposes of testing, but this has also been done to allow extensibility as the interface can be extended and implemented by a different class to allow the functionality of the program to be extended, while maintaining compatibility with the current system following the open-close principle.

Another area in which the system supports extensibility is through it's upgrade system, using the decorator pattern it allows a player to be extended with any number of upgrades, as well as the creation of further upgrades. This upgrade system works in tandem with the ability system to allow upgrades to change how effective abilities are based on their damage type. Although the upgrade system currently does not do this, it does also support the ability to modify variables of the player such as health, lives, max health, magic etc. which could be implemented at a future date.

5.3.2.3 Reusability

The system is also designed with reusability in mind, ensuring every component is reusable, and duplication is minimal. One example of this is in the ability system, which is the same for all entities both player and enemy. This allows us to get significant millage out of just a small number of abilities.

Another way the system makes good use of the resources it has is through it's encounter system. As an encounter can comprise of multiple enemies, allowing us to make use of enemies that the player would otherwise be very strong against in later encounters. Furthermore encounters at most stages are randomised, providing variety to each run, along with the randomisation of what abilities are purchasable in the shop.

5.3.3 Abilities

5.3.4 Upgrades

5.3.5 Enemies

5.3.6 Player

5.3.7 Telemetry

5.3.8 Time

5.3.9 Settings

5.3.10 Game Mangement

5.3.11 User Interface

6 Data Flow Schema

TODO: TC

7 Telemetry Schema

TODO: TC

8 Game Design

TODO: WF

9 Initial Evaluation

TODO: WF

10 Ethical and Legal Analysis

TODO: WF

11 Testing Architecture

Our prototype consists of five automated tests using the JUnit framework (version X.X.X). These automated tests all test the behaviour of the TelemetryListener class, which processes telemetry events as they happen - fundamental to the core objective of our program. Details on how to run these can be found in our Deployment Guide (deployment_guide.pdf).

We have implemented three unit tests - these test that the userID, sessionID and timestamp fields of events are validated and handled correctly. By supplying varying fields of Event objects with erroneous data, we have asserted that exceptions are handled by verifying the contents of the error message output to console. When supplying all of these fields with typical data, we have asserted that no exception handling occurs, since no exceptions were thrown in these cases.

We have also implemented two integration tests - these test that telemetry events are written to file if the user has enabled telemetry, and vice versa. By invoking the listener's method on a valid Event object, we have asserted that the file's length increased immediately after the event occurred if telemetry was enabled, and that the file's length did not change if telemetry was disabled. Any data read from or written to file in our test was done using temporary files, to avoid cluttering in the production files.

// TO DO - Discuss manual tests once done

12 Software Bill of Materials

TODO: EA

13 Sprint 2 Priorities

TODO: TC

14 Management

TODO: KK