Technical Design Document (TDD)

[Introduction](#_b1jd59vhbdtz)

[Document outline](#_ojbjwfl12cs4)

[CRC cards](#_72trjrv80ut9)

[Architecture](#_yn7g5aglwc3a)

[Technology](#_rteb78q76r41)

[Sprint 1 Report](#_u6sdzhjbytqv)

# 

# 

# Introduction

A Description of each feature[[1]](#footnote-1)/user story[[2]](#footnote-2) is held in Game Design Document (GDD). The GDD describes *what* will be built.

The Technical Design Document (TDD) describes *how* it will be built - it describes the detailed design of the system and its components. The TDD will grow and evolve over the lifetime of the project. But at a minimum, the TDD should contain detailed design for all the features to be implemented in the upcoming sprint.

**NB**: Before starting to write the TDD, ensure all questions are answered about *what* will be built.

The TDD includes:

* A complete set of CRC cards (these are **physical** cards)
* The architecture (UML class diagram)
* Features - For each feature in the system (use the same names as the feature in the GDD), give a breakdown into a set of tasks that are clear enough for a developer to implement without asking any questions.

# Document outline

The TDD should have the following structure:

* CRC cards
* Architecture
* Technology
* Sprint 1 report[[3]](#footnote-3)
* Sprint 2 report
* Sprint 3 report

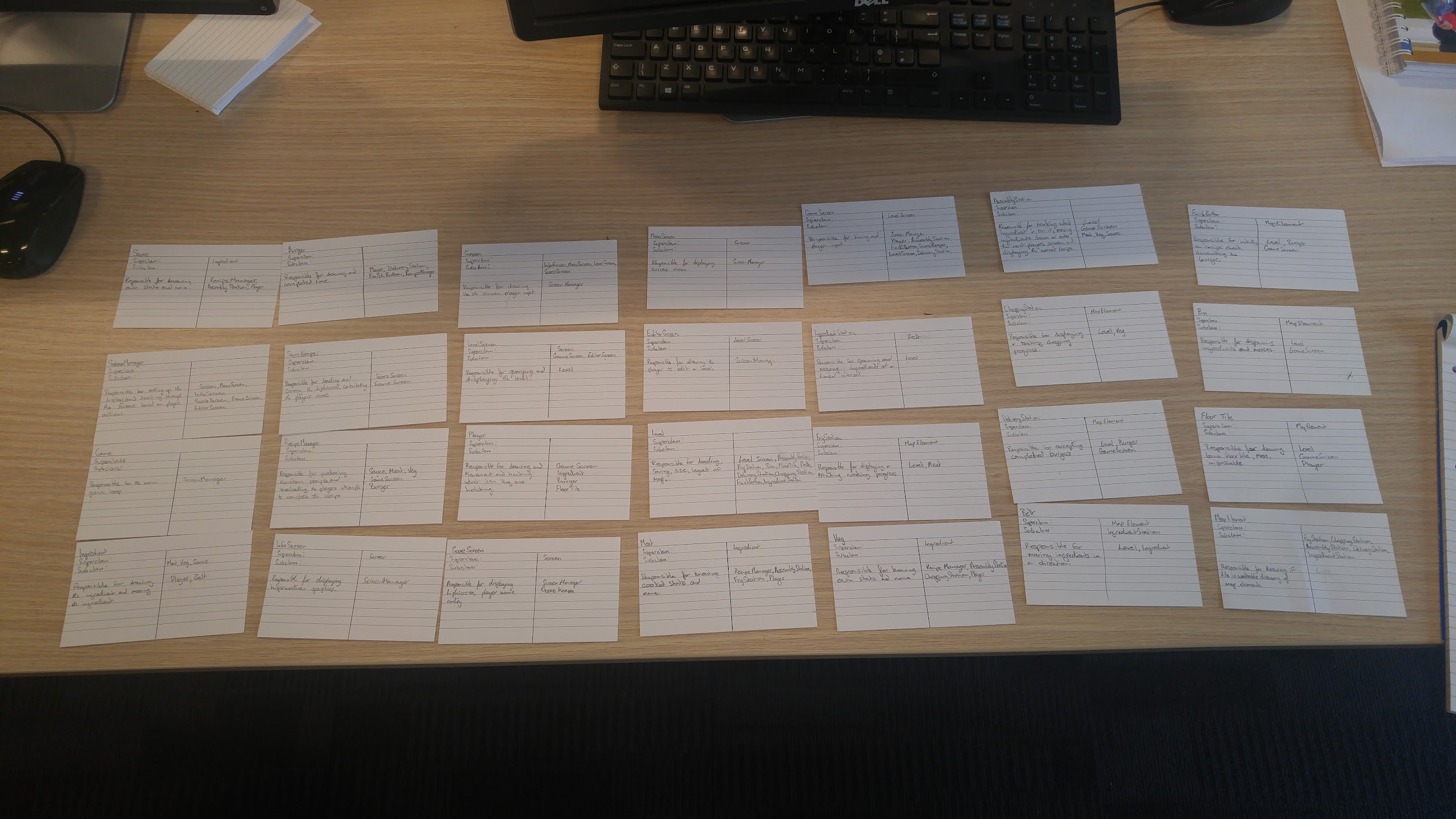
Note use of style for headings:

* Heading 1 for high level heading e.g. CRC Cards, Architecture, Features
* Heading 2 for the name of a feature
* Heading 3 for the name of a task

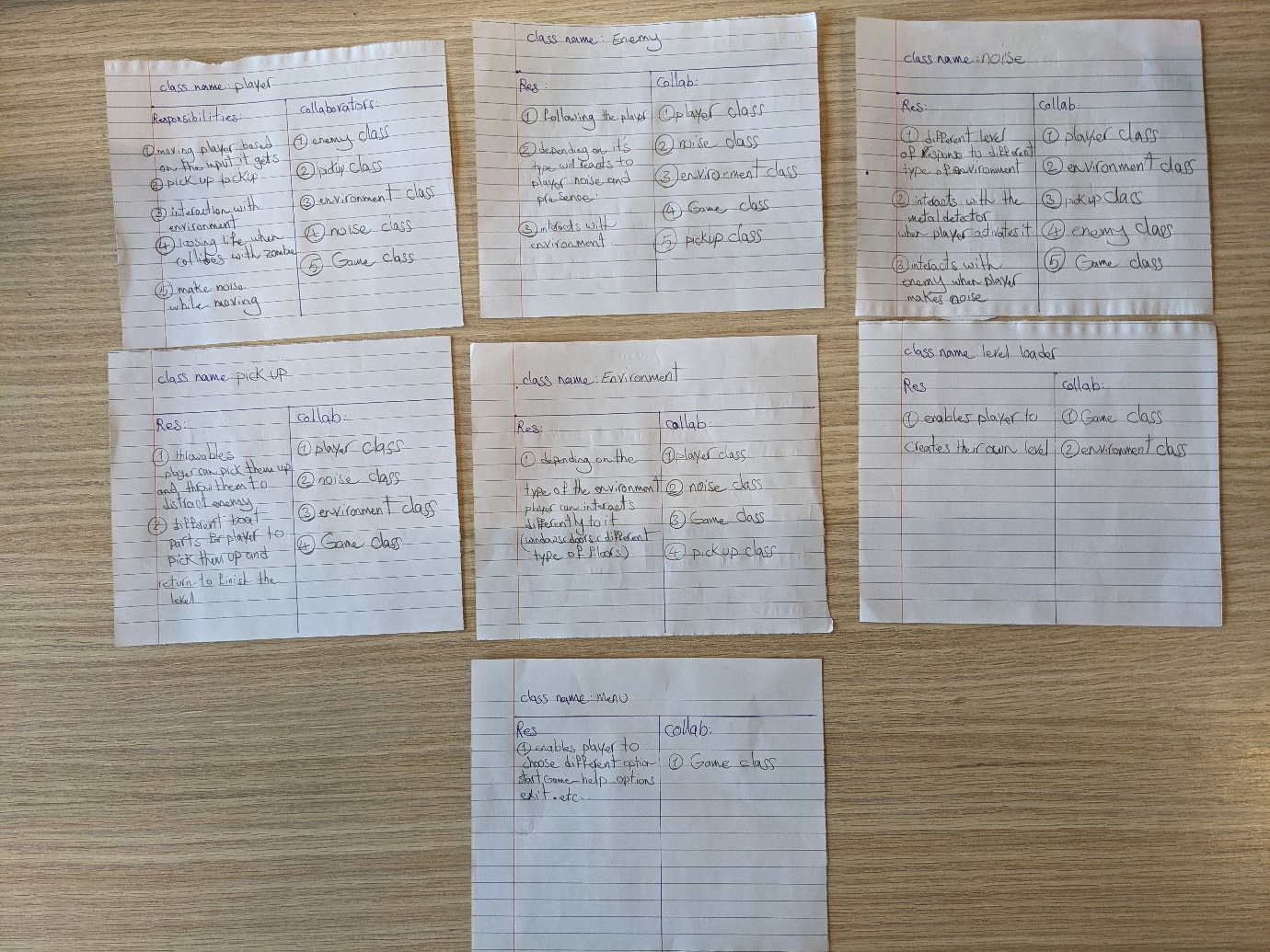
# CRC cards

<http://agilemodeling.com/artifacts/crcModel.htm>

<https://creately.com/diagram/iycufdln/Card%20Game%20CRC%20cards>

Example 1[[4]](#footnote-4): [link to set of photos of physical cards]

Example 2: [Sample CRC Cards from space shooter game](https://drive.google.com/open?id=1N4SngEh5h9hJ9A3MzwrtNymd-3zEqk-PEcChGHfy6WA)



# Architecture

UML class diagram showing relationships between classes identified with CRC cards. There should be two class diagrams. The first will show the initial work planned for sprint 1 only.

The second class diagram is to be produced at the end of the project and will show the entire architecture.

# Technology

This section should document all aspects related to the technology used in your project and will contain subheadings provided below. You will add to this section of the TDD as you encounter new technology throughout the life of the project. Include technologies not used but seriously considered.

**Research**

Begin by outlining the rationale for your technology choices (i.e. justify your choices) and any research you did in relation to support libraries or other aspects related to development. For example, if you picked one SFML GUI library over another, justify that decision. We suggest you are economical in your use of time in relation to the research element (e.g. 30 minutes max on any one particular aspect).

**Installation**

Explain how any dependencies/libraries are installed in your project. Provide URLs where the software can be downloaded.

**Technical achievement**

This section should be completed at the end of the project by each team member. You should highlight one part of the project that you felt was particularly challenging but managed to complete. This could be a pathfinding algorithm, a rendering technique, a complex game mechanic etc. Include relevant code snippets in your discussion.

# Sprint 1 Report

**Summary of planned work**

|  |  |  |
| --- | --- | --- |
| **Features and tasks** | **Time (Hours)** | **Team Member** |
| **Feature 1: Moving Player** |  |  |
| Task 1: Render a rectangle |  |  |
| Task 2: Setup input handler |  |  |
| Task 3: Create game loop |  |  |
| Task 4: Allow Player to modify speed |  |  |
| **Feature 2: Enemy** |  |  |
| Task 1: Render the Enemy |  |  |
| Task 2: Setup Collision with Player |  |  |
| **Feature 3: Environment** |  |  |
| Task 1: Render the object |  |  |
| Task 2: Setup Collision with player and enemies |  |  |
| Task 3: Setup children to inherit from the above |  |  |
| **Feature 4: Noise** |  |  |
| Task 1: Draw the Noisebar |  |  |
| Task 2: Allow Noise to be added |  |  |
| Task 3: Make the bar vary based on noise level |  |  |
| Task 4: Setup a radius that detects nearby zombies based on noise level |  |  |
| **Feature 5: Pickups** |  |  |
| Task 1: Draw Pickups on the screen |  |  |
| Task 2: Setup Collision with Player |  |  |
| Task 3: Allow certain Pickups to be thrown |  |  |
| Task 4: Setup Collision between objects and environment |  |  |

**Work completed**

This is the last heading relating to the sprint report and this section must be completed at the end of the sprint. Copy the table from the planned work section and add an extra ‘Complete’ column as indicated below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Features and tasks** | **Time  Estimate** | **Time Actual** | **Team Member** | **Complete** |
| **Feature 1: Horizontal movement of player character** | **7** |  |  | N |
| Task 1: Render a rectangle | 1 |  | John | Y |
| Task 2: Setup input handler | 1 |  | Jane | Y |
| Task 3: Setup game entities | 2 |  | John | N |
| Task 4: Create game loop | 1 |  | Jane | Y |
| Task 5: Give player maximum velocity & acceleration | 2 |  | Jane | Y |
| **Feature 2: Move ball** | **6** |  |  | Y |
| Task 1: Render the ball | 2 |  | John | Y |
| etc... | ... |  | ... |  |

This indicates which tasks (and therefore features) are complete (Yes or No). You should provide an explanation below the table for each task that has not been completed.

## Videos

At the end of each sprint, create a video no longer than 1 minute that showcases the new game features you have completed. Videos should be edited to provide context for the viewer. For example, include concise captions with short and simple text. Upload your video to your project folder on google drive - prefix your video with the name of the sprint, e.g. Sprint1, Sprint2 etc. See next page for suggestions on video capture/editing software.

Suggestions for video capture/editing

Video Capture

Microsoft Stream (screen capture with cloud save and download options)

<https://www.microsoft.com/en-us/microsoft-365/microsoft-stream>

Note! You will need to be logged into your Office 365 Windows account. Stream saves captured video in a format called webm. Ideally you will want to download your video as an mp4 file for editing. A simple work around is to rename the .webm extension to .mp4

How to trim video in Stream:

<https://docs.microsoft.com/en-us/stream/portal-edit-trim>

Video Editing

Windows 10 includes a free Video Editor app which you can use to edit MP4 files (type ‘Video Editor’ in the Windows search box)

1. A feature is a unit of functionality which adds value to the user's experience playing the game. [↑](#footnote-ref-1)
2. User-stories are a concept from mainstream agile software development, and describes a specific goal they wish to achieve using the software. We prefer to use the term ‘feature’ in game development as the user’s only goal is often to “be entertained”. [↑](#footnote-ref-2)
3. Each sprint report describes features, tasks, planned work and completed work. [↑](#footnote-ref-3)
4. No need to recreate the cards digitally in the document. Ensure photos are clear and legible [↑](#footnote-ref-4)