Table of Contents:

1. Introduction and Project Ideas

1.1 Introduction

The goal of this project is to create a 2D arcade-style game, suitable for an audience of 12 and over, within 20 weeks. We began our project having everyone within the group brainstorm possible 2D game ideas, whilst studying published popular 2D games to improve our understanding. We further discussed the reasons for the game's success (what the positives and negatives of each game were), how well it fits with our target audience as well as how we could implement some of the key features of the games. We then explored all of the games that were suggested and voted on our favourite ideas. Finally, after explaining our top 3 ideas to the lecturer, we narrowed down our possible options and decided on our final game idea for our project. This will be further elaborated in the next sections.

1.2 Initial Ideas

The following were some of our initial proposals when deciding on a final plan. These are important to mention as we drew much of our inspirations, basic mechanics and design features from their examples.

-1.2.1 Maze game

Our initial game idea was a maze puzzle game, where the maze would be procedurally generated and the goal of the player would be to traverse it whilst collecting items to solve puzzles. We had thoughts of adding a hack and slash approach or a turn-based combat system.

Eventually, we discarded this game idea, due to it being too easy and lacklustre both from a gaming (and programming perspective), hence not being suitable for our task and target audience. We appreciated the maze idea, however, it wasn't a suitable focal point for a game. Despite this, we did take a liking to the hack and slash feature/idea, which we delve into later on.

Similar existing games that influenced this idea:

- Pac Man
- Scary Maze Game
- World's Hardest Game

Key features:

- Procedurally generated maze
- Collectible items for puzzle solving
- Hack and slash
- Turn-based combat

-1.2.2 Platformer

This game idea was a no brainer, with so many 2D games being platformers. The particular idea we had was a side-scrolling, jumping puzzle game with perhaps simplistic combat. This was definitely a strong contender to be our final game idea. However, this form of platformer is highly dependent on good map/level design rather than complex code.

Similar existing games that influenced this idea:

- Super Mario
- Hollow Knight

Key Features:

- Jumping Puzzle
- Side-Scrolling

-1.2.3 Racing Car game

A continuous running car game where the player controls a fast-moving vehicle down a motorway. The goal for the player is to achieve the longest distance and gain as many points as possible before getting caught by the enemy bot. The story we had for the game was that the player is racing down a motorway and encounters police officers who are the enemy bots in the game. The player must avoid obstacles and frequently refuel by picking up fuel items to ensure that a safe distance is kept between the player and the enemy bot. The game ends whenever the enemy bot catches the players.

Similar existing games that influenced this idea:

- Subway surfers
- Temple Run
- Hill Climb Racing

Key Features:

- Time or resource constraints (Must pick up fuel to continue, etc)
- Being chased by enemy AI
- Special items can be picked up

We decided not to go for this idea because we wanted a gratifying ending where a user has a clear end goal.

-1.2.4 Music/Rhythm game

Players perform actions to the rhythm or tune of music, which can take the form of acting as if you are playing the music, beating enemies in a rhythmic fashion, or dancing to it

Similar existing games that influenced this idea:

- Geometry Dash
- Guitar Hero/ Piano tiles
- Beat Saber
- OSU

Key Features:

- Each level is choreographed to the rhythm of its song
- Visual and audible cues help the player time inputs
- Combo streak for multiplied score

We decided against this idea because we feel that most of our time developing the game would be spent on level design and creating a decent soundtrack for the game, rather than demonstrating our software development skills. Although our team sees a good soundtrack as a desirable goal, we would prefer to have the soundtrack and core game mechanics separated entirely, to firstly improve our focus on creating a functional and enjoyable game.

-1.2.5 Falling block game

This concept was inspired by tetris, and in a way city builders - building blocks fall from the sky and the player orientates them to gradually build up a city. As the player completes buildings, they get points - this also stops them from developing the area later on. While this could have been fun to play, it is too simple a design to build into an interesting 2D game project and we feel that there is not much innovative freedom in this idea.

Similar existing games that influenced this idea:

- Tetris
- Cities Skylines

Key Features:

- Random pieces are generated and fall from the top of the screen
- Different buildings need to be built in different areas/level

1.3 Final Game Idea - Dungeon Crawler Concept

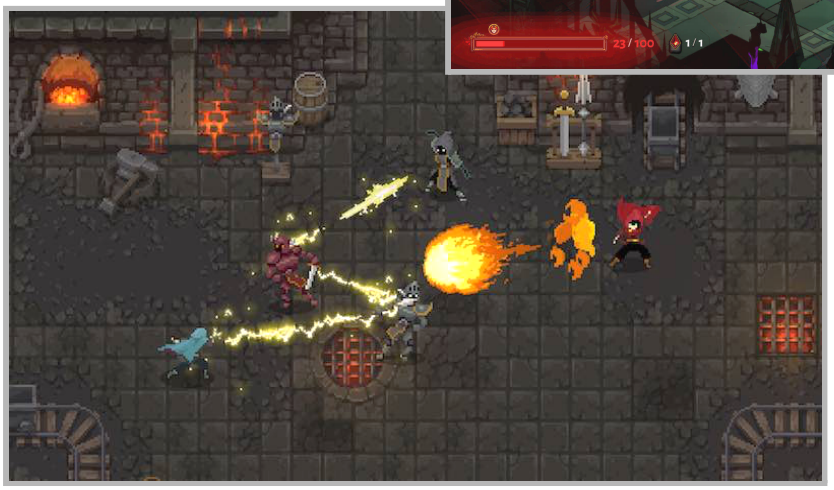
The idea is to have a single player, top-down view game that revolves around the player defeating all the enemies in each level to progress to the next. The end goal is to defeat the final boss. The player is meant to die often, in order to gain more “permanent” upgrades and understand the game mechanics by experimenting with different strategies. The main game mechanics are focused on “temporary” upgrades that the player selects at the end of each level. These upgrades can stack/combine with each other to produce different “builds”/strategies e.g melee build, tanky build, ranged build etc. Temporary upgrades are upgrades you lose when you die and permanent upgrades are kept throughout the playthrough. Understanding the game mechanics will be key to progressing through the game effectively and greatly encourages replayability.

After consulting our lecturer in the labs, we reached a consensus that a dungeon crawler would be the most ideal choice for our group. Our group members are all familiar with this style of game and the traditional game mechanics found in this genre. This makes a dungeon crawler the most practical idea. We also noticed that LibGDX has a few examples of similar games to a dungeon crawler in their documentation, which made it more enticing to commit to.

We also decided to put our own spin on our interpretation of the genre by considering features from the games previously mentioned (1.2.1-1.2.5). We looked at rhythm games and decided that timing on attacks could have an influence on their damage or effect given a visual or audible cue. We also thought about introducing puzzles on behalf of maze games and platformers. We have also taken inspiration from other large games with many complex features, but we have to be realistic given the constraints of time.

Similar existing games that influenced this idea:

- Hades
- Rogue Legacy
- Binding of Isaac
- Pixel Dungeon
- Wizard of Legend
- Hollow Knight
- Dead Cells



Reference images:
https://www.npr.org/2020/11/15/934539471/life-is-hellish-enough-thats-why-i-had-to-stop-playing-hades?t=1637925006220
https://gadgets.ndtv.com/games/reviews/wizard-of-legend-nintendo-switch-review-1852034

2. Game Design

2.1 Game Design & Principles

-2.1.1 Objectives

We will use some simple goals (may not be explicitly written) to immerse the player into the game, and let them advance until larger, more difficult goals become clear.

- The game will have different levels that the player will need to defeat in order to reach the objective/boss level.
- The meta objective is to defeat the boss (and so the game) with each weapon.
- The player's objective is to become stronger, more skilled and knowledgeable in order to achieve the meta goal of beating the game.
    - An implicit initial objective for the player is to die/lose in order to become stronger and progress further in the next “run”/attempt.
- The enemies become more challenging the closer the player gets to the boss level, increasing the difficulty. The implicit objective is to defeat the enemy without dying.

-2.1.2 Beginner Friendly

In order to engage players with the game in the first place, it must be beginner friendly

- Each new ability or mechanic in the game should be introduced one by one, when they are relevant, so the player is not overwhelmed.
- The environment surrounding the player when a mechanic is introduced should demonstrate to them how it works, and let them experiment with it if they want to. By this method, we reduce the need for text-based instruction, or standard tutorials which can easily be slow and boring to play, even if helpful.
- There will be a training room in the main lobby, where the player can return to view and test skills, gear, and game mechanics.

-2.1.3 Replay Friendly

The entire design of this game is built around replaying the dungeon, so it is therefore clearly important to make our game fun to replay. Towards this goal there are several different principles we will implement:

- Each playthrough will be different through a combination of player choice from a variety of choices supporting different play styles, and some random variation (RNG factors) throughout the game such as available abilities, types of enemies and a randomised map for each level.
- There will be many abilities, and other mechanics, that could invoke different styles of gameplay that will give players a lot to figure out over multiple playthroughs. Since they only get one new ability per level so they can experiment with one new mechanic at a time; they won’t be easily overwhelmed.
- The player gradually learns more about the story through progressing further and further each playthrough.

-2.1.4 Rewarding

Related to replayability, the game will reward players for performing well, whether it be quickly beating a level, or figuring out a good combination of abilities.

- At a basic level, the player should feel rewarded for defeating enemies, so they will provide currency or potentially other items when defeated.
- Enemies should also be rewarding and satisfying to fight. The player must have an impact on the fight beyond simply dealing damage. This could take the form of dodging attacks, comboing attacks, employing strategies and using the environment for an advantage.
- Each enemy can drop currency when defeated, the value increasing with difficulty, and mini bosses dropping a different type of currency that can be used to unlock permanent skills.

-2.1.5 Control

When a player feels they do not have control over the character, they do not have fun because they do not feel responsible for negative events (e.g. taking damage because the character didn't move), and undeserving of positive ones (e.g. the character stopping when they did not mean to, and avoiding an attack). Keeping this in mind, we should give them as precise control as possible.

- The player should use a keyboard and mouse to control their character using traditional keyboard bindings. (W, A, S, D, LMB, RMB, etc)
- At all times when the character could be damaged, the player must have complete control, and be able to avoid it using knowledge and experience or audible and visual cues to detect incoming attacks.
    - Not necessarily by dodging during a fight - also perceptive things like noticing signs of a trap on the ground
- Actions such as key presses have responsive, visible and intended effect(s) after they are performed - any animations for starting to move or attack an enemy must be short and clear.
- Range of enemy attacks should correspond to their graphical representation (e.g. length of a weapon when swung).
- "Hitboxes" should be accurate when ranged enemies fire projectiles at the player.

-2.1.6 Accessibility

We find accessibility to be an important feature of our game - every person should be able to enjoy playing our game, not just to appeal to a wide audience, but also to ensure that nobody feels excluded.

- There will be multiple difficulty levels, so that if the game is too challenging for a player and they are not enjoying it, they can make it easier; if it is too easy and a player is bored, they can make it harder.
- The graphics design will show clear contrast of colour and shape between the player, enemies, and other elements on screen. This is helpful for visually impaired people, and in general makes potential reaction speed faster.
    - Colours could be customisable for colourblind players, so that they can clearly see the intended differences in colour and shape of enemies and other elements.
- Our game will support different input keys for actions inside the game, for those unable to use a mouse, for example. LibGDX also has a library that supports gamepad, which makes it even more accessible for users that want to use different hardware for controls.

-2.1.7 Story

The story is set in a fantasy setting. The player has been trapped in a labyrinth. Hostile creatures block their path to freedom. They finally decide to escape, but they are killed by a creature, and discover that they are truly trapped in the labyrinth for eternity - they cannot escape even in death, they are cursed to exist within the labyrinth. They continue exploring the labyrinth, searching for any way to rid themself of the curse and escape, even if it means they have to die an indefinite number of times to find the right path.

The player develops skills and abilities over his time in the labyrinth, allowing them to defeat more enemies and travel further into the labyrinth than they did in their previous lives. After defeating the first "boss", the player learns that they must defeat the "final boss" to be freed from the labyrinth.

Textual cutscenes will be played at key points within the game to reveal more of the story to the player, such as the introduction, after each death, after defeating a boss, and upon defeating the game.

2.2 Game Features and Gameplay Overview

- The game is single player only
- The game takes place in a series of randomised previously made maps (with exceptions such as lobby/boss level, which aren't randomised)
- Each map represents a single level
- There is a set number of levels (will be less than 30)
- Every (e.g 5) N of levels there is a mini-boss room
- The final boss is in the last level
- The player needs to defeat all the enemies in each level to progress to the next level
- The player will get a reward for beating the level
- There will be 1-3 "doors" that will unlock after beating a level
- Each "door" shows the reward you get when you beat the next level

- Rewards range from “temporary” upgrades to currency
- The player will be able to use different attacks and can “combo” (sequence of attacks)
- There should be a main menu and pause menu
- Each playthrough of the game is a “run”, the run ends when the player dies
- The player keeps a single (permanent) type of currency whenever they die
- The player returns to the beginning of the game when they die
- The player can spend (permanent) type of currency for permanent upgrades before/after each run
- The player selects a weapon at the start of each “run”
- The player gets “temporary” upgrades (upgrades that are lost when they die/the run ends)
- The upgrades the player gets can stack/combine and be used together
- The player is expected to die often (expected to fail many “runs”), get permanent upgrades and learn game mechanics before beating the game

-2.2.1 Enemies

The enemies who the player has to defeat throughout the game are of different types.
Some of the enemy types are:

- Normal (Well balanced)
- Fast
- Slow but powerful
- Armoured
- Have different attack patterns
- Ranged
- Have features e.g stun, poison etc.
- Boss type enemies

So for example, the player will have to deal more damage to an armoured enemy than a fast enemy.
Enemies will also have different “affinities”. For example, armour may protect against physical damage but not magical attacks.
This will add more variety to the method of defeating each enemy, making the game more interesting as it won't have the same repetitive patterns. If the player masters defeating only one type, it may get easy and boring, so different enemy behaviours are essential.
There will also be mini bosses dotted throughout the game and a main boss to beat at the end, to complete the game. Mini bosses and the main boss will be more difficult to defeat as they may have better stats (eg. higher attack power) and will have different attack patterns to regular enemies.

-2.2.2 Levels

The player will begin in a starting lobby with an introduction, a store for permanent upgrades, and a training area to test different weapons and abilities. This area will be the same for each run of the game, but following levels will be semi-randomised.
Levels could contain the following:

- Traps
- Bonus chests
- Locked doors
- Impassable terrain
- Dark and bright areas
- Boss rooms
- Rest areas
- Temporary item stores
- Differing themes and music
- Easter eggs that tell the game’s story

Each standard level can randomly select a map, from a pool of pre-made maps and every N level will contain a boss room. The player will be able to change levels by travelling through one of 3 doors. Each door will give the player a certain ability depending on which the player selects, and the level they enter will be designed around that ability - e.g. if they can use their ability to move quickly, they might have to dodge or catch quickly moving enemies.

-2.2.3 Temporary Upgrades

With each new level the player progresses to, they will gain a new ability, which will help them defeat enemies, such as damage absorption, dashing temporarily, or a multi-attack. However, they will only have a limited choice of abilities at any given time, encouraging them to vary their playstyle when facing different enemies. These abilities will stack/combine/be used simultaneously. These upgrades will be lost upon death/defeat of the final boss. Here’s what they should do:

- An advantage against specific enemies or in certain environments.
- Increase player’s damage output
- Increase player’s survivability

-2.2.4 Permanent Upgrades

Every time the player reaches the lobby by dying, the player will lose all temporary currencies and upgrades. They will be left with only the permanent currency that can be used to buy permanent upgrades at a shop in the lobby. These upgrades will give the player more utility in their next playthrough, perhaps allowing them to explore more of the levels, have extra lives before they reset, or assist in combat.

-2.2.5 Currency

In the game there are two types of currencies - temporary and permanent.

- Permanent currency is kept after a player dies - used for permanent upgrades, skills, etc. between runs in a lobby shop
- Temporary currency is lost when a player dies - only being used for temporary items during a single run
- Smaller enemies will drop temporary currency and bosses will drop permanent currency
- The player can convert permanent currency into temporary before each run

-2.2.6 Bosses

To have more varied gameplay, bosses will be included - they add spikes of difficulty, meaning the player has competing interests, in wanting to preserve health, but also needing to fight as many enemies as possible to get currency and abilities to beat bosses

- Bosses will combine features of enemies in previous levels, testing the player's skill in fighting them
- The player is not expected to beat the first boss they encounter, as they have not prepared for it.
- Beating the final boss is the goal of the game (beating the game)
- Bosses can summon other enemies to help them fight.

2.3 Game Rules

-2.3.3 Constitutive rules

- Player starts with 1 life and 100% health at the beginning of each run.
- Enemy spawns with 100% health
- Enemies are spawned in each level (except lobby level)
- Enemy health increases with each level, and they become harder to kill.
- Enemies become smarter and are more challenging to fight with each new level.
- Player spawns at the start of each level, by the door/path they came in.
- Player cannot attack inside the attack cooldown.
- Player can exchange currencies for temporary or permanent upgrades at shops.
- When the player kills an enemy the enemy drops loot.
- If the player loses all their health, they go back to the start (lobby).
- If the player beats the final boss, they are given permanent rewards and go back to the start (lobby).
- The player can only change weapons when they are in the lobby.
- Player cannot leave the map / visit areas unintended for gameplay.

-2.3.3 Operational rules

- The player can attack enemies under given conditions (e.g: within range, attack ready) to inflict damage and/or debuffs
    - Different types of attack can be performed with the same weapon
    - The attack type and weapon will affect these conditions (e.g: heavy attack has longer cooldown, ranged weapons have long range)
- Enemies will inflict damage to the player if they come within a certain range of the enemy and the enemy's attack is ready
    - Enemies can select different attacks suitable to the situation - such as switching to a ranged weapon if the player runs away.
- The player can move in valid directions using WASD.
- The player collects currency when the character touches it.
- The player can choose which upgrades to have active at any one time.
- The player can use upgraded abilities to augment attacks / movement.
- The player can change game settings after pausing during gameplay
- The player can attack in different directions.
- The player can perform different types of attacks.

-2.3.3 Implicit rules

- The player's character follows the controls given by the player through the keyboard and mouse.
- The player should use abilities from upgrades to gain an advantage over, and ultimately defeat enemies.
- The player should gain knowledge and experience about the labyrinth after each death, using this to understand enemies and game mechanics to find their way through the levels more quickly on subsequent runs.
- The player should be perceptive, looking for hidden details that may help them piece together the story, or pathways that may be taken after certain upgrades make terrain passable for the player.
- The player should use different types of attacks depending on how effective they are against specific enemies.
- The player can choose to save & quit at any point during gameplay.
- The game shouldn't be intensive on system resources.
- The player should choose their upgrades carefully.

2.4 Tables

(Limited space, so a small number of exemplar enemies, weapons and upgrades are displayed)

Enemy	Type	Health	Base Damage	Movement Speed	Attack Pattern
Slimes	Normal	100	20	30	Pounces at player
Hounds	Fast	150	20	50	Rushes to player and pounds to attack
Giants	Slow but powerful, Armoured	300	40	15	Throws boulders in several places, swings bat at player if they are within range
Skeletons	Armoured, Long Ranged	250	25	25	Shoots arrows at player
Dragon Boss	Poison, Armoured	500	50 +Poison: 5 every 3 secs if player is poisoned	25	Breathes fire ranged attack Swings tail at player, Swipes claws at player repeatedly

Weapon	Base Damage Unit	Attacks Per Second	Attack Pattern
BroadSword	100	2	3 Sword Swings
Bow	60	1.25	Single Timed Shot
Dual Daggers	30	4	6 Consecutive Attacks

Temporary Upgrades	Description
Speed Potion	Increase Movement Speed
Health Potion	Increase Max Health of Player
Critical Potion	Adds critical chance to Player weapon
Deflect Potion	Adds ability to deflect ranged attacks with Player weapon

Permanent Upgrades	Description
Dash	Press a button to quickly move in a given direction
Backstab	Increase damage dealt to enemies backs
Heal Reward	Regain some health upon beating a level

3. Software Engineering

3.1 Assumptions

-3.1.1 Player

- **A1**: The player is familiar with and able to use basic keyboard controls.
- **A2**: The player can read English.
- **A3**: The player is age 12 or over.

-3.1.2 Hardware

- **A5**: The computer in use meets the minimum requirements to run Windows 10.
- **A6**: The user has an appropriate version of Java installed.
- **A7**: The user has a standard keyboard and mouse connected to the computer.
- **A8**: The user has a high resolution display and speakers connected to the computer.
- **A9**: The aspect ratio of the user's display is 16:9.
- **A10**: The computer has enough storage space to install the game and hold save data for one playthrough.

3.2 Software Requirements

We will use specific language in this section to describe our requirements for the game. These requirements can be broken into two categories: "Shall" and "Should" requirements. This describes whether a requirement is necessary, for example, core game mechanics and features, or if it is simply desirable, for example, graphical requirements, music, or animations. These will be listed as either functional or non-functional requirements.

-3.2.1 Functional Requirements

Basic

- **F1**: The game shall allow player movement using (W, A, S, D keyboard keys)
- **F2**: The game shall allow player attacks using (LEFT CLICK, RIGHT CLICK with mouse)
- **F3**: The game shall allow the player to aim attacks using the mouse
- **F4**:The game shall allow the player to access the menu using ESC key
- **F5**: The player shall not be able to move outside the map
- **F6**: The player shall be able to be damaged by enemy attacks
- **F7**: The player shall be able to damage and kill enemies
- **F8**: The player shall be able to gain temporary upgrades
- **F9**: The player shall be able to select a weapon
- **F10**: The game shall have multiple weapons
- **F11**: Different weapons shall have different attack patterns
- **F12**: The game shall remove dead enemies
- **F13**: The game shall have boss levels
- **F14**: The game shall be able to spawn enemies in levels
- **F15**: The enemies shall have pathfinding AI
- **F16**: The enemies shall attack the player
- **F17**: The enemies shall have different attack patterns
- **F18**: The game shall have shops
    - **F18.1**: Player shall be able to purchase permanent upgrades using permanent currency in the lobby shop
    - **F18.2**: Player shall be able to purchase temporary upgrades using temporary currency in the level shops
- **F19**: The game shall restart from lobby upon player death
- **F20**: The game shall restart from lobby upon defeating the game
- **F21**: The game shall reward the player with rare permanent currency upon defeating the boss
- **F22**: The game shall reset player's temporary upgrades and temporary currency
- **F23**: The game shall save player's progress at the start of every level
- **F24**: The game shall save player's permanent upgrades and permanent currency

- **F26**: The game shall display necessary information to the player through GUI
    - **F26.1**: Current health
    - **F26.2**: Current currency balance

    - **F26.3**: Current health of damaged enemy
    - **F26.4**: Current upgrades in possession
    - **F26.5**: Damage the player deals to the enemy
    - **F26.6**: Damage the enemy deals to the player
    - **F26.7**: In-game shop
- **F27**: The game shall display the correct animations
    - **F27.1**: Player movements
    - **F27.2**: Player attacks
    - **F27.3**: Player death
    - **F27.4**: Enemy movements
    - **F27.5**: Enemy attacks
    - **F27.6**: Enemy death
- **F28**: The game shall handle level/map progressions
    - **F28.1**: Shall handle level completion
    - **F28.2**: Shall handle switching and loading levels
    - **F28.3**: Shall display cutscene
    - **F28.4**: Shall handle skipping cutscene
- **F29**: The game shall have varying level of difficulties
- **F30**: The game shall output correct sounds
    - **F30.1**: Player attacks
    - **F30.2**: Player movement
    - **F30.3**: Player dealt damage
    - **F30.4**: Player death
    - **F30.5**: Enemy attacks
    - **F30.6**: Enemy movement
    - **F30.7**: Enemy dealt damage
    - **F30.8**: Enemy death
    - **F30.9**: Level music
- **F31**: The game shall allow the player to interact with in-game world using the E key
- **F32**: The game shall allow the player to interact with the GUI using the mouse
- **F33**: The game shall have doors/paths for player to progress from each room
- **F34**: The player shall be able to stack/combine certain upgrades
- **F35**: The game shall not allow the player to stack/combine mutually exclusive upgrades
- **F36**: The game shall have a lobby level

Preferable

- **F37**: The game shall have menus
    - **F37.1**: Main menu
    - **F37.2**: Save/Load menu
    - **F37.3**: Options menu
    - **F37.4**: Pause menu
- **F38**: The game should have a minimum of 10 different levels
- **F39**: The game should have a minimum of 10 different upgrades
- **F40**: The game should have a minimum of 3 different weapons
- **F41**: The game should have a minimum of 5 different enemies
- **F42**: The player shall be able to change the game setting in the options menu

-3.2.2 Non-Functional Requirements

- **NF1** : The game should display at adjustable resolutions
- **NF2** : The game shall run at 60 FPS
- **NF3** : The game shall load into the main menu within 20 seconds
- **NF4** : The game shall load into next levels within 10 seconds
- **NF5** : The game shall not crash (freeze up or close prematurely) during at least 98% of gameplay
- **NF6** : The game shall not be affected by unexpected key presses
- **NF7** : The game shall save player progress without fail at least 99% of the time
- **NF8** : The game should encrypt the ‘save’ file to prevent player editing their progress
- **NF9** : The game shall be written in Java (JDK 15+)
- **NF10** : The game shall run on SCC Ubuntu lab images
- **NF11** : The game shall use libraries of Java SDK and libGDX
- **NF12** : The game shall respond correctly to keys that serve a purpose in the game
- **NF13** : The game shall run without errors or setbacks on a system with the minimum requirements to run windows 10

3.3 Acceptance Tests

Functional Requirements

ID	Description	Input	Expected Output
F1	The game shall allow player movement using (W, A, S, D keyboard keys)	Start level, press WASD keys	Player character should move in correct directions, respective to keys
F2	The game shall allow player attacks using (LEFT CLICK, RIGHT CLICK with mouse)	Start level, click left mouse, click right mouse	Player character should attack according to respective mouse click
F3	The game shall allow the player to aim attacks using the mouse	Start level, aim mouse cursor and attack using left/right click	Player character should attack in correct direction towards cursor position
F4	The game shall allow the player to access the pause menu using ESC key	Start level, press ESC key	Game should display pause menu
F5	The player shall not be able to move outside the map	Start level, move player character towards walls/boundary edges of map	Player character should move no further than wall/ boundary edges in game
F6	The player shall be able to be damaged by enemy attacks	Start non-lobby level, allow enemy to attack player character	Player character should be damaged, health bar GUI should decrease
F7	The player shall be able to damage and kill enemies	Start non-lobby level, attack enemy	Enemy health and health-bar GUI should decrease, and die when empty
F8	The player shall be able to gain temporary upgrades	Start level, select a temporary upgrade via GUI	Player gains temporary upgrade chosen, ability should change according to upgrade
F9	The player shall be able to select a weapon	Start lobby level, go to weapon area in lobby and select a weapon via popup GUI	Player weapon changes in appearance according to selected weapon
F10	The game shall have multiple weapons	Start lobby level, view weapons in weapon area and popup GUI for weapons selection	Multiple weapons for player to select from
F11	Different weapons shall have different attack patterns	Start lobby level, go to weapon area, select each weapon and attack with each	Show different attack patterns/animations for each weapon
F12	The game shall remove dead enemies	Start level, kill an enemy	Enemy should disappear from screen soon after death
F13	The game shall have boss levels	Start boss level	Boss should spawn in boss level
F14	The game shall be able to spawn enemies in levels	Start non-lobby level	Enemies should start spawning
F15	The enemies shall have pathfinding AI	Start non-lobby level, move to different areas of map	Enemies should path-find towards player
F16	The enemies shall attack the player	Start non-lobby level, move close to enemies and wait	Enemies should attack player
F17	The enemies shall have different attack patterns	Start non-lobby level, move close to different looking enemies	Different looking enemies should have different attack patterns
F18	The game shall have shops		
ID	**Description**	**Input**	**Expected Output**
F18.1	Player shall be able to purchase permanent upgrades using permanent currency in the lobby shop	Start lobby level, go to “permanent” shop, open shop GUI and try purchase permanent upgrade	Permanent upgrades should be only bought (if player has enough currency), removed from the shop and will be displayed in player GUI/HUD
F18.2	Player shall be able to purchase temporary upgrades using temporary currency in the level shops	Start non-lobby level, go to “temporary” shop, open shop GUI and try purchase permanent upgrade	Temporary upgrades should be only bought (if player has enough currency), removed from the shop and will be displayed in player GUI/HUD
F19	The game shall restart from lobby upon player death	Start non-lobby level, allow enemy to kill player character	Player should die once health reaches 0, after cutscene, player is re-spawned in lobby level
F20	The game shall restart from lobby upon defeating the game	Start final boss level, attack the boss	Once boss health is 0, boss dies, after cutscene, player is respawned in lobby level
F21	The game shall reward the player with rare permanent currency upon defeating the boss	Start final boss level, attack the boss	Once boss health is 0, boss dies, after cutscene, player is respawned in lobby level with increase in rare permanent currency
F22	The game shall be able to reset player’s temporary upgrades and temporary currency		
F21.1	Upon player’s death	Start non-lobby level, allow enemy to kill player character	Player dies and re-spawns in lobby level without any temporary upgrades/currency
F22.2	Upon final boss’s death	Start final boss level, attack the boss	Once boss health is 0, boss dies, Player re-spawns in lobby level without any temporary upgrades/currency
F23	The game shall save player’s progress at the start of every level	Progress to a level, quit game anytime before moving to next level	Upon relaunching game, player’s progress at the start of the level saved, is loaded in
F24	The game shall save player’s all upgrades and all currency at the start of every level	Gain upgrades/currency by completing levels with them as rewards. Progress to next level and quit game	Upon relaunching game, player’s upgrades/currency at the start of the level saved, is loaded in
F26	The game shall display necessary information to the player through GUI		
F26.1	Current health	Launch new/saved game	Should be visible on bottom left of screen
F26.2	Current currency balance	Launch new/saved game	Should be visible on bottom right of screen
F26.3	Current health of damaged enemy	Start non-lobby level, attack enemy	Health of damaged enemy should appear above their heads
F26.4	Current upgrades in possession	Launch new/saved game	Icons of upgrades in possession should be visible on left border of screen
F26.5	Damage the player deals to the enemy	Start non-lobby level, attack enemy	Damage numbers should pop up next to enemy
F26.6	Damage the enemy deals to the player	Start non-lobby level, allow enemy to attack player character	Damage numbers should pop up next to player character
F26.7	In-game shop	Start lobby/shop level, go to shop area	Pop-up GUI from shop should be visible

ID	Description	Input	Expected Output
F27	The game shall display the correct animations		
F27.1	Player movements	Move player character	Correct animation plays corresponding to movement direction
F27.2	Player attacks	Attack using player character	Correct animation plays corresponding to attack pattern and direction
F27.3	Player death	Start non-lobby level, allow enemy to kill player character	Correct animation plays corresponding to player death
F27.4	Enemy movements	Start non-lobby level, allow enemy to move to player	Correct animation plays corresponding to movement direction
F27.5	Enemy attacks	Start non-lobby level, allow enemy to attack player character	Correct animation plays corresponding to attack pattern and direction
F27.6	Enemy death	Start non-lobby level, kill enemy	Correct animation plays corresponding to enemy death
F28	The game shall handle level/map progressions		
F28.1	Shall handle level completion	Beat a level	Path/Door to next room appears
F28.2	Shall handle switching and loading levels	Beat a level and try to progress to next via path/door (test for all levels) and loading of lobby level via player death/boss defeat	Next level loads correctly
F28.3	Shall display cutscene	Progress through all cutscenes in-between levels	Each cutscene loads correctly and in order
F28.4	Shall handle skipping cutscene	Progress through cutscene in-between levels and attempt to skip them via GUI/key press	Cutscene should be skipped and next level should load
F29	The game shall have varying level of difficulties	Start lobby level, move to path/door to start “run”/next level	GUI pop ups allowing selection of difficulty
F30	The game shall output correct sounds		
F30.1	Player attacks	Attack using player character	Correct sound plays corresponding to attack pattern
F30.2	Player movement	Move player character	Correct sound plays corresponding to movement of character
F30.3	Player is dealt damage	Start non-lobby level, attack enemy	Correct sound plays responding to damage
F30.4	Player death	Start non-lobby level, allow enemy to kill player character	Correct sound plays responding to player death
F30.5	Enemy attacks	Start non-lobby level, allow enemy to attack player character	Correct sound plays corresponding to attack pattern
F30.6	Enemy movement	Start non-lobby level, allow enemy to move to player	Correct sound plays corresponding to movement of enemy
F30.7	Enemy is dealt damage	Start non-lobby level, allow enemy to attack player character	Correct sound plays responding to damage

ID	Description	Input	Expected Output
F30.8	Enemy death	Start non-lobby level, kill enemy	Correct sound plays responding to enemy death
F30.9	Level music	Start level	Music should play in loop
F31	The game shall allow the player to interact with in-game world using the E key	Start level, go to interactive world object (e.g shop, door/path et.c) and press E	Correct interaction should take place, GUI should open for selection related world objects
F32	The game shall allow the player to interact with the GUI using the mouse	(Launch game try to navigate menus) and (Start level, try to open pop up GUIs e.g shops)	Player can navigate correctly through GUIs
F33	The game shall have doors/paths for player to progress from each room	Start level, defeat level and try to enter doors/paths	Player character and next level will be loaded in
F34	The player shall be able to stack/combine certain upgrades	Play through game, select compatible upgrades	Upgrades abilities should stack/combine/used together
F35	The game shall not allow the player to stack/ combine mutually exclusive upgrades	Play through game, select incompatible upgrades	Incompatible/mutually exclusive upgrades, new upgrade shall replace old upgrade
F36	The game shall have a lobby level	Start lobby level	Lobby level loads
F37	The game shall have menus		
F37.1	Main menu	(Launch game) or (Quit from in-game Pause menu)	Main menu should display, with navigation buttons to Save/Load menu and Options menu
F37.2	Save/Load menu	Launch game, navigate from Main menu	Save/Load menu should display, with 3 save files buttons (empty/used)
F37.3	Options menu	(Launch game, navigate from Main menu) or (In-game Pause menu)	Options menu should display, with setting options
F38	The game should have a minimum of 10 different levels	Start game, progress through game, count number of different levels	There should be at least 10 different levels
F39	The game should have a minimum of 10 different upgrades	Start game, progress through game, count number of different upgrades	There should be at least 10 different upgrades to select and use from
F40	The game should have a minimum of 3 different weapons	Start lobby level, go to weapon area in lobby and view and select weapons via popup GUI, attack with weapon	There should be at least 3 different weapons to select and use from
F41	The game should have a minimum of 5 different enemies	Start game, progress through game, count number of different enemies	There should be at least 5 different enemies

Non-Functional Requirements

ID	Requirement	Input	Expected Output
NF1	The game should display at adjustable resolutions	- Launch game on lab PC. - Resize the game's window to a different resolution. - Attempt F11 or maximise window	The game should adjust to the new desired resolution.
NF2	The game shall run at 60 FPS	- Play games with maximum theoretical number of enemies spawned in	The game should display the FPS to be at least 60.
NF3	The game shall load into the main menu within 20 seconds	- Measure how long it takes the game to reach the main menu from launch.	The game should load into the main menu within 20 seconds.
NF4	The game shall load into next levels within 10 seconds	- Launch the game on a lab PC. - Complete the first level. - Measure how long it takes the game to load the next level.	The game should load the next level within 10 seconds.
NF5	The game shall not crash (freeze up or close prematurely) during at least 98% of gameplay	- Launch the game. - Play through the game. - Count number of crashes	The game shouldn't crash during gameplay.
NF6	The game shall not be affected by unexpected key presses	- While the game is running, select random keys outside of the key presses expected from the user.	The game shouldn't be affected by unexpected key presses.
NF7	The game shall save player progress without fail at least 99% of the time	- Play through game - Check that player progress is saved when the run is over.	The game should save the player's progress 99% of the time.
NF8	The game should encrypt the 'save' file to prevent player editing their progress	- The game is saved.	Save data should be automatically encrypted before being written to a save file.
NF9	The game shall be written in Java (JDK 15+)	- Development principle	The game will be written on a lab machine or a personal device running the same version of java.
NF10	The game shall run on an SCC Ubuntu lab images	- Run the game on a Ubuntu lab PC.	The game should run on SCC Ubuntu lab images without any issues.
NF11	The game shall use libraries of Java SDK and libGDX	- Development principle	The game will be written with only Java SDK's standard libraries and libGDX.

4. Implementation

4.1 First Term

-Week 4

- **In the workshop** we first met most of our team mates and then all contributed various game ideas and discussed their positives and negatives as well as some examples of games we could relate our game ideas to. We compiled these on paper, later expanded in this document
- **In our meetings** we finalised our game idea and made a decision on which one of our previous suggestions we were going to go after, making sure everyone has the same concept of what the game would be like and gave some suggestions on what implementations there should be. We wrote down the key concepts and features of the game as well the aim of the game so that we could gain approval in our next workshop on our chosen game.

-Week 5

- **In the workshop** we presented our game idea to the TA/lecturer and gained approval to go ahead with the idea. Then we started looking at and noted down the design of the game such as aspects like music, graphics and game mechanics.
- **In our meetings** we started the design report and looked at the design repost examples provided. We analysed them and chose our favourite layout and characteristics of each that we liked.

-Week 6

- **In the workshop** we finally met every member of our team in person and continued with our design report which was on google docs so we could all work on it simultaneously. After making our contents page, we each worked on different sections whilst occasionally checking and discussing ideas and wording with each other.
- **In our meetings** we worked on the report individually and as a team. It was really beneficial checking with each other what features of the game was a necessity and which we could choose to not include. It was also useful in deciding what our functional and non-functional requirements were.

-Week 7

- **In the workshop** we started to finalise this report, working on section 3 and 4. We also discussed with the lecturer on how our design report was looking and he gave us positive feedback on our non-functional requirements, and advised us to include risks we could face that could disrupt our schedule.
- **In our meeting** we completed section 3 and section 4, we used asana for the activity network and google sheets for the gantt chart. We further polished up on our full report and added missing contents in the previous sections. We discussed our approach to assigning work and concluded it was not as efficient as it could be. Continuing hereon we will use Trello.

4.2 Plan For Development

-4.2.1 Delivery Schedule

Programming				
ID	Milestone	Deliverables	Dependencies (Chained)	Week of Delivery
P1	Program entity-component system	Java class files providing core functionality for entity-component management. Storing entities with components of data.	None	8
P2	Program rendering system	Java class files providing core functionality for a rendering queue. To draw/render updated graphics per loop.	None	8
P3	Program event queue system	Java class files providing core functionality for an event queue. To communicate between different systems.	None	8
P4	Program core functionality of the player's character	Java class files providing core functionality for the player. Includes: status, movement, attacks, interaction, death.	P1, P2, P3, GS1, GS2	9
P4.5	Program player character	Java class files providing implementation of functionality for the player.	P4,	9
P5	Program core functionality of levels	Java class files providing core functionality for levels. Such as wall boundaries, impassable terrain, traps and doors.	P1, P2, P3, GS1, GS2	9
P5.5	Program testing level	Java class files providing functionality and logic for the testing level, such that core elements of the game can be tested.	P5	9
P6	Program core functionality of enemies	Java class files providing core functionality for enemies. Such as movement, spawning, death, collision, attack and AI.	P1, P2, P3, GS1, GS2	9
P7	Program core functionality of the GUI / HUD	Java class files providing core functionality for a GUI / HUD. They will ensure that the GUI can obtain all required information and update the display accordingly.	P4	10
P8	Program core functionality of the player combat system	Java class files providing core functionality for the player's combat. This will handle hitboxes and attacks.	P4, P6, L2	10
P9	Program core functionality of the enemy combat system	Java class files providing core functionality for enemy combat. This includes: enemy attack patterns, targeting system.	P6, L2	10
P10	Program extra functionality of movement system	Java class files providing extra functionality for movement for player and enemies. Such as collision.	P4	11

P11	Program core functionality of player weapons	Java class files providing core functionality for player weapons. Such as status and attack patterns.	P4, GS1, GS2, (GS4)	11
P12	Program core functionality of player upgrades	Java class files providing core functionality for player upgrades. Such as weapon upgrades, player upgrades, temporary upgrades.	P4	11
P13	Program core functionality of the shopping system	Java class files providing core functionality for the shopping system. Such as currencies, transactions, items and GUI.	P4, P5, P7,	12
P14	Program core functionality of textual cutscenes system	Java class files providing core functionality for the textual cutscene system. Such as cutscene appearance, textual and flow of cutscene.	P1 - P7	12
P15	Program core functionality of the game control flow system	Java class files providing core functionality for the game control flow system. Such as progression/flow of levels, player death and beating boss.	P5, P6, P7	12
P16	Program core functionality of the game menu system	Java class files providing core functionality for the game menu system. Such as a main menu, popup/pause menu, saving progress, exiting and restarting.	P7	12
P17	Program each level	Java class files providing core functionality for each level in the game. This will provide logic for wall positions, impassable terrain, enemy spawn positions, and doors for each level.	P5, P13	14
P18	Program each enemy	Java class files providing core functionality for each enemy type in the game. This will depict the different stats, affinities and behaviours of each enemy type.	P6, P9	14
P19	Program each weapon	Java class files providing core functionality for each weapon that the player has access to.	P9, P11	15
P20	Program each upgrade	Java class files providing core functionality for player upgrades and skills.	P12, P13, P15	15
P21	Program shops	Java class files providing core functionality for shops and trading.	P13	16
P22	Program GUI/HUD	Java class files implementing a final GUI/HUD	P7	16
P23	Program game progression	Java class files that implement play progression.	P17, P15, P12	16
P24	Program each cutscene	Java class files that implement more complex cutscenes.	P22	17
P25	Program menus	Java class files that implement the final menu system.	P14, P7	17
P26	Program saving game progression	Java class files that allows saving of certain game states and player progression	P25	17

Content Design				
ID	Task	Deliverables	Dependencies (Chained)	Week of Delivery
L1	Design levels (at least 10)	Documentation describing levels that will be later implemented	None	8
L2	Design player & enemies (at least 5)	Documentation describing enemies that will be later implemented	L1	9
L3	Design weapons (at least 3)	Documentation describing weapons that will be later implemented	L2	10
L4	Design upgrades (at least 20)	Documentation describing upgrades that will be later implemented	L2	10
L5	Write a script for each cutscene (at least 4)	Script for game cutscene, mainly for staging	L4	11

Graphics & Sound				
ID	Task	Deliverables	Dependencies (Chained)	Week of Delivery
GS1	Create simple placeholder textures to support core development.	Simple set of graphical assets, purely for prototyping.	None	8
GS2	Create simple placeholder sounds to support core development.	Simple set of audio assets, purely for prototyping.	None	8
GS3	Create pixel art assets of character	Simple graphic design of the player, in the form of an image / gif.	None	9
GS4	Create pixel art assets of enemies	Simple graphic design of enemies, in the form of an image / gif.	None	9
GS5	Create pixel art texture assets for levels/maps	Simple graphic design of levels, in the form of an image / gif.	GS3, GS4, L1	10
GS6	Create pixel art assets for special effects	Simple graphic design of special effects.	L3, L4	11
GS7	Create sound assets	.WAV sound effect assets for players, enemies, levels and special effects	L3, L4	13

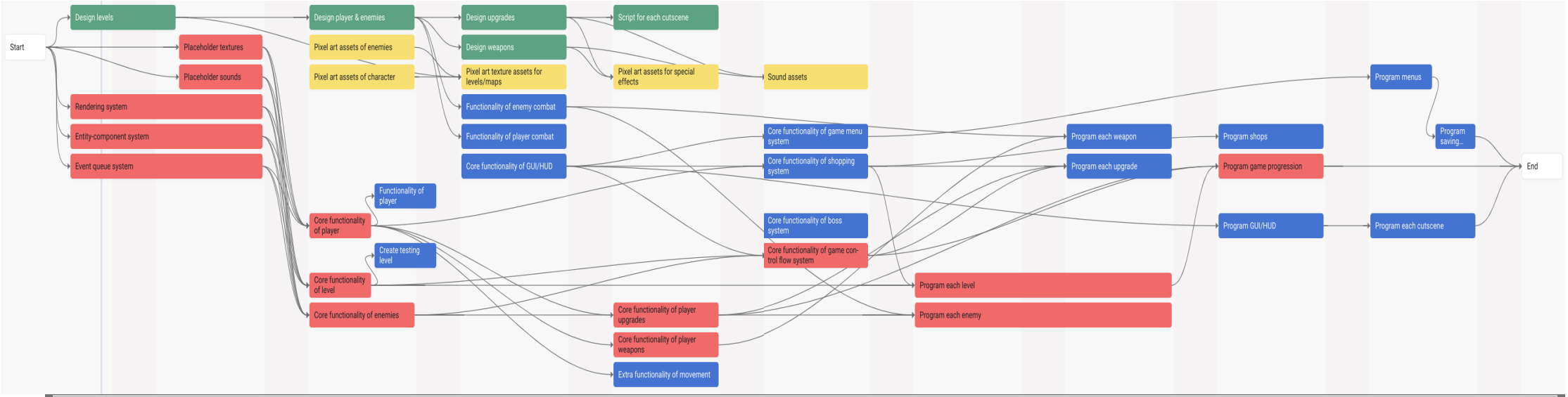
-4.2.2 Risks

Throughout the weeks of development in terms 1 & 2, there is significant risk that events will occur that could have a substantial effect on our team meeting the deadlines we have set. We have prepared this section to identify likely risks, and discuss ways in which we can reduce the impact of such events on our ability to follow our deadlines.

- Our main risk is that a member of the team leaves before development is finished, or is absent for a portion of development. We have prepared for this by scheduling tasks pertaining to three different sections of our requirements, based on potential difficulty and complexity of implementation. This, alongside allocating specific tasks to members of our team for each week, and monitoring dependencies, allows us to closely watch our progress and change each team member's focus dynamically throughout development.

- Another risk is that it may take more time than expected to meet a deadline for a deliverable. To mitigate this, we have ordered tasks such that we accomplish "Basic" requirements first, which allow the game to function, whereas "Preferable" requirements have a lower priority; the tasks pertaining to which can be pushed back to a later deadline, enabling the game to reach a playable state prior to the addition of more ambitious features.

4.3 Activity Network



Key: Content Design, Graphics & Sound, Programming, Critical Path

Task nodes for Critical Path: Start -> Placeholder textures, Placeholder sounds, Rendering system, Entity-component system, Event queue system -> Core functionality of player, Core functionality of level, Core functionality of enemies, Core functionality of player upgrades, Core functionality of player weapons, Program each level, Program each enemy, Program game progression -> End.

4.4 Gantt Chart