Changes made to the project plan:

I started working on the game very late, so to compensate to add as many features as possible that would make it look and feel like a game, I decided that I would be pushing back key concepts of my game. This led to removing the entire concept of items and the sacrifice mechanic within my game. This sadly removed the identify of a game and left me down with a prototype for a 2d top-down shooter. Additionally, due to some unforeseen circumstances in the final week I had to scrap the whole idea for a unique boss fight that would go in with the loop of the game if I wanted to meet deadline, I had conceptual ideas to what the boss fight would have been but non could be started to be designed as there simply was no time to add on these. Aside from these two major concepts which I had to take off my game ruining its identity I worked in order I found fitting for my game development process. Plan was to begin with incorporating a character sprite first, which I found online on a open guide, I wanted to replace this character and its animations as I was planning to end the project but decided there is no time for that and left it in as a clear example of managing movement animations and fitting representation to how big the character would have been if I were to add a custom one made by myself in the game. Following that in my development run I choose to work on simple ranged enemies, I designed them and made them so they turn towards the player every tick of the game as well as shoot in his direction with perfect accuracy but slow projectiles as to add the ability for the player to dodge without the use of any further mechanics which would make a simple enemy like this not a treat. Proceeding forward I stayed with my idea of making a procedurally generated map for each level. The plan shown in the map design from before was kept and nothing was changed. Each level by default stats the same with an open square where the player can freely move and then proceed to open and make unique corridors for movement in each level. There is no controller to how small a room can be but there is a hard limit to how big they can. An idea of expansion was adding a controller for sizes of rooms scaling with difficulty and level in progression through the game however that was knocked off as I could not think of a favourable algorithm in short term to balance that aspect of the game.
Proceeding on I choose to make unique weapons for the player. I chose to make 3 weapons, pistol, submachine gun, and a sniper weapon, all of which have their own properties and shooting formats and damage. As I had decided I would not be implementing items, I chose to add scaling to the player that would be justifiable in its values and keep the game healthy with its difficulty, unless playing on the highest difficulty which is only made to show severe scaling of the game loop. Additionally, for the same reason as weapon damage scaling, I added a passive heal to the player's health every "x" amount of rounds which was only fair as there was an idea to make weak "life steal" mechanics or "drain" item upgrades which would have sustained the player further. However, it is very clear to any game developer that `life steal` as a concept is very difficulty to balance evenly and the only ways to implement it without damaging the game is to limit it heavily so the healing the player received would only matter at standard difficulties as on the highest the scaling of enemies is way simply unfair, as it would suggest playing on a game mode made for only extreme challenges.
Additionally, in the beginning I wanted to make the game have more customizability within

its level generation however that was knocked off as it was a small extra idea, I wanted to add which I felt as if would add more unique experience. Trap and small obstacles around were removed as an idea for implementation as with how the map was being generated it was really difficulty to incorporate them without using set methods which I found less exciting for the player such as random pre-defined spawn points for such obstacles on the playable field.
Another change done to the game was the reward. Instead of the game being completed once you beat stage 30 (floor 3 – level 10), I moved it, so the player beats the game each 10 levels but with the ability to continue to the next floor and proceed to play the game. If he chooses to end the game and start over again, he is granted the ability to do so via a very basic UI displayed for beating the game.

Which design patterns you used, why you choose these ones and where these can be found in the code.

- Singleton – seen as to how I manage general global variables and classes. This was used to simplify the sharing of data in between different functions and Game Objects. It is present within `GameManager` and `Stats` objects as they are the key ones needed for the game to operate with through game loops.
- Unity's Update method – used to track the player mostly and used for specific raycasting verifying that for example a melee enemy has reached the player.
- Unity's Components – used to attach key functionality feature to elements of the game.
- Event Queue – used for example to validate if the stage has been cleared.
- Object Pool – Commonly used within the game. It was needed to manage the number of objects my game would create for example with the procedural map generation.
- State – Used primarily with enemies and scanning if the level was completed with all the enemies being defeated.
-

Where you used the required games concepts and where this can be found in the code

Level generation – procedural level generation was one of the key concepts I wanted to execute as it is a requirement for a game such as mine to be validate within its respectful genre.
Game loop – allowing for replay ability with different weapons and constantly changing experience as the key factor that affected it in this case was the custom maps.

Fitting to a dungeon crawler / roguelike genre I implemented 3 different types of weapon for the player to use with their own properties and ways of working. This allows to enhance the game loop as the player is provided more options to play with and attempt beating the game.

The security requirements considered and how you implemented these.

The game did not have any need for heavy security requirements to be implemented by game developer standpoint. It is a single player top-down shooter that doesn't require an input as in: `email`, `name`, `password` or any other personal information from the player. In terms of protecting the game loop, I did not consider adding limitations to cheats such as the very simple ejection and value manipulator `CheatEngine` like software because there was no fear from the player cheating to get ahead in a certain game loop. That does not provide fear to the game as with games with scaling algorithms of enemies and the player itself with those cheats you can reach limits which were not reachable by default. This would not cause errors unless you overloaded the engine's limit which was the only security risk that was looked at as it was the only "cheat" to the game that would have been found unhealthy for the game.

Anything you would do different if you were to tackle this project again.

I think my flow of work with this project was on point for the time that I had. Researching Unity and C# on the spot was done in good batches of time then followed by my work after which usually would be smooth unless a major error provided a slow-down. The only real issue for me was my limitation of time which made the entire project unmanageable. This sadly I had very little control of and was realistically the only reason I could not even assess myself for this report in this section. I believe my work plan was correct and the only real development change I could have had was how I took off with player first rather than choosing to work on a slightly bigger scale with the `GameManager` and then breaking down components of it. Potentially an extra change that would have been probably very healthy for development is having a closer contact with the adviser for the module to determine the limits of what I can use and what I cannot. If I were to take this project in my own time and push it as a passion project rather than work, I need to do I would have chosen to use certain well-designed assets for the look of the game.

Your testing strategy and the results of the testing.

The key testing, I did were:

1. Compatibility testing – This took a long time as there were many issues I encountered with the final state of my project. I had a lot of User Interface issues

which took a long time to fix as I was not understanding of certain parameters within Unity. These tests were carried over the entire development process as well for small components and prototype features which I added that usually failed but provided enough information as to how they can implement and adjusted so they fit the build.

2. Performance testing – by far the least time consuming one as the game at its final project had very little things with it. The game was well optimised when it was tested on my own computer and seemed to be well reinforced in basic manners.

Other manners of tests:

1. During Development, each component was tested separately in a prototype prefab that was created for it. I did this to improve my understanding of certain concepts within Unity more so than improve the development process and this was very helpful in the long run as I understood how to make the game very well procedurally ran whilst using a very clean method for generating the game.
2. Friend Play Testing – Friend suggested that the cursor could be changed from the dark red colour it had to a bright green colour, so it is in contrast with the enemies and their projectile to not lead to confusion. Additionally, mentioned an issue with the game that hid some enemies when they spawned at the exiting point. This was a rare bug that was encountered and was looked at when mentioned. Aside from that the general feedback was positive and the game seemed to have been easy enough to pick up and play as on his first play through he had no difficulties managing with the game pace and difficulty for a single clear of the game at a normal difficulty.
3. Play Tested by myself – I played through the game on all difficulties, to ensure the enemy scaling worked as intended. Made sure that all assets were used correctly, and all the features implemented. This test was successful as the plans for each feature testing were very clear in my own tracker and it clearly worked well for it. Additionally, with each of the tests I carried on my project I used each run's high and low points to determine how I should alter the scaling, so it feels fair and to the player and it does not provide too many difficulties. Doing this helped make a health game loop and lead to very good results with how the game was altered from the primary idea.