February

16/02
Module Introduction.

18/02
Started to research about the technologies that would be used in the project and assigning roles to the group members.

March

02/03
Game was planned to be developed using Unreal Engine, After a meeting with the group members it was decided that the game would be developed in Java 8 , so I started researching on how a game could be developed in Java.

09/03
Started researching about how a Game loop works in java, found different approaches for a game loop, the first one using the Timer library in java, the second Using Thread, and the third one was a more sophisticated version of the game loop using Thread it has a better performance compared to the others and has a feature to set the frequency that the game would be updated and the frequency that images are drawn on the screen.

16/03
Started a local version of the project, Game loop was implemented, Started working on the Abstract Class “Game Object”, User input Class and a player character Class. The Game Object Abstract Class was quite an interesting concept, every “Object” in the game share some common attributes such as X position, Y position, Speed , also most objects would need an Update/”tick” method and a Render/”Draw” method, having this abstract class would give us the ability to reuse code without having to copy and paste it everywhere in the project.

27/03
Started putting the pieces together, introduced the Handler class, this class is responsible for storing every game object into different structures such as Linked Lists and Array Lists, this approach would improve the overall game performance, making it possible to make update and render calls to different groups of game objects, Some updates were made to the Game Panel class, which is where the game itself runs, Some changes to this class still needs to be made.

30/03
Started working on the Character Inventory Class, this class was developed using some Java Swing libraries and is part of the GUI, includes panels, labels, images, scrollable panels, text area and receives mouse input.

April

O6/04
Started working on the game mechanics :
Collision Detection
Player movement (Gravity, Acceleration, Deceleration, Jumping)
Character State Machine (handles all the character states in the game)

13/04
Developed the animation process, loading images into memory, getting sub-images from an image, the animation system is quite sophisticated , it was designed in a way that would reduce the amount of code needed and the amount of images , it uses the concept of sprite sheets, it was also designed to offer the developers the ability to play different animations at different speed (“play rate”) , this process is called every time the character state changes.

20/04 The TileMapper Class
This was definitely one of the most challenging weeks, I started to plan how we could generate the levels for the game in an efficient way, luckily the platformer genre is very popular and there are thousands of assets (tile sets) that can be downloaded from the internet, after a long research I’ve found a software (Tiled) where we could import these tile sets , quickly design a level and export this level data as a json file, the software also let us design the level using layers, which was very handy to keep each element of the map in a category to be later on stored in the appropriate data structure, that was essential for the system to work with good performance, It was very challenging to work with the json data, the exported json file consists on an JsonArray, so I had to read this file on Java, break down/filter all th
e information necessary to build the level, extract the data , convert values e.g((long) to int, extract json object from json array), map all this data into appropriated data structures, then develop the process to generate the level. The level generation process is a bit complex, the system load an image (the tile set) this image contains every tile that is used in the level, so I have to break down that image into sub-images (tiles) and store all those images in an array, the json file contains the level data (which is an array of numbers) these numbers are the indexes of those tiles in the tiles array, the map data is stored in an 2d array the index of each value stored in that array is used to calculate where the image will be draw (the row Index is a multiplier used to calculate the Y position/vector of that tile and the Column index is the multiplier used to calculate the X position/vector).The json file also contains data representing the width and height of the level, the width and height of the tiles for that specific level.
The system was working as intended but there was a performance issue, if we have a large level with thousands of tiles, the game performance would get compromised, to fix that I have to add some logic that would only render the tiles that were within a certain offset, this offset is calculated based on where the player character is and the size of the screen.

27/04
Side Scrolling camera implemented, Added enemy class based on the player character class, I felt that I could have done an Abstract class for Characters and then created the player character and enemy classes from that class, but I did not have enough time for it.
Added the Zenith class (particles that flies and bounces on the level)

Implemented the interaction between the enemies and the zeniths where the enemies can be infected by the zeniths
Added the projectile class.
Added the player ability to shoot projectiles.
Created custom Thread classes to handle events independently from the game loop process, which adds the concept of multi-threading to the project.
Added a health bar to the GUI.
Implemented the “game over “ sequence.

May

04/05
Could not work on the project this week, overloaded with other projects.

11/05
Minor bug fixes and adjustments.