Collider

Functional Specification

Name: Ore Ibikunle

Student Number: 15351216

Finishing Date: 26/11/2018

Table of Contents

1. Introduction:

1.1 Overview:

Collider is a 2-Dimensional space shooting game. The objective of the game is to manoeuvre across space avoiding and destroying different types of asteroids. The system to be developed will include a well-designed GUI and a database to store players high scores.

1.2 Purpose:

The purpose of this project is for me to develop an understanding of design principles and algorithms used in game development.

1.3 Glossary:

- **Sprite** – a computer graphic which can be moved on-screen and otherwise manipulated by a single entity.

- **Sprite Sheet** – is an image that consists of several smaller images (**sprites**) and/or animations.

- **Canvas** – a component that is represented by a blank rectangular area of screen onto which the application can draw or from which the application can trap input events from the user.

- **Vector** – is a quantity that has magnitude and direction. It is used to determine the position of one point in space relative to another.

- **Collision** – an instance of one moving object striking violently against another.

- **Collision Detection** – is the computational problem of detecting the intersection of two or more objects.

- **Intersection** – the point at which two or more things intersect.

- **One-Dimensional Newtonian Formula** – is used to create an accurate reaction when two or more objects collide.

- **Velocity** – is the speed of an object and the direction in which it is travelling.

- **Trajectory** – is the path followed by a projectile flying or an object moving under the action of given forces.

- **Two-Dimensional** – appearing to have length and breadth but no depth.

- **Screen Tearing** – occurs when what is being graphically displayed from a game is not in sync with the display's refresh rate.

- **Lag** – is a time delay between a player's action and the game's reaction to that input.

- **Spawning** – is the live creation of an object.

2. General Description:

2.1 Product / System Functions:

Below is the list of the main functions of the game to be designed. This is a preliminary list and is open to additions should I think of anything worth adding:

- Canvas Creation
- GUI Menu
- User I/O Interaction
- Storage of player scores in a Database.
- Realistic Movement of Objects
- Detecting collision between different type of objects.

2.2 User Characteristics and Objectives:

The intended system is designed for users with low-end computers and for those who use their computers a lot daily.

The objective of the game is for the player to manoeuvre a spaceship across space avoiding two types of asteroids, larger and smaller ones. Larger asteroids can collide with the smaller ones, changing the asteroids trajectory, making it harder for the player to avoid. The player can shoot down the smaller asteroids but not the larger ones, the game becomes more difficult as the player continues. The score will be determined by the number of smaller asteroids destroyed by the user.

2.3 Operational Scenarios:

2.3.1. Player Opens the Application:

- **Current State:**

  The system consists of no players because the player has not started the game. The system is waiting for the player to start the game.

- **Informal Scenario:**

  The player has selected to look at the controls of the game.

2.3.2. Player Selects to View Controls on Menu:

- **Current State:**

  The system now displays the controls to the player and how to play the game.

- **Informal Scenario:**

  The player has now read how to play the game and goes back to the main menu where they select to start the game.

2.3.3. Player Manoeuvres the Spaceship:

- **Current State:**

  The player has started the game and the system state will now consist of the player controlling the spaceship and trying to avoid the larger asteroids and destroy the smaller asteroids.

- **Informal Scenario:**

  The player will repeat this process until they either quit the game or are destroyed by an asteroid.

2.3.4. Player Destroys Asteroid:

- **Current State:**

  The player shoots and destroys an asteroid, the system state has now changed, the asteroid they destroyed is no longer being displayed by the Canvas and the system increments the players score by one.

- **Informal Scenario:**

  The player continues in their attempt to destroy more smaller asteroids and avoid collision from the larger asteroids.

2.3.5. Players Score Increases:

- **Current State:**

  As the player score increases the system displays the asteroids moving at a quicker pace.

- **Informal Scenario:**

  The players concentration and hand movement increase as they attempt to destroy smaller asteroids or avoid both type of asteroids.

2.3.6. Larger Asteroid Collides with Smaller Asteroid:

- **Current State:**

  The system state changes again, causing the trajectory and velocity of the smaller asteroid to change and its acceleration to increase.

- **Informal Scenario:**

  The player notices the change in speed of the smaller asteroid and the quickly attempts to avoid it or destroy it depending on their position on the canvas.

2.3.7. Player Destroyed by Asteroid:

- **Current State:**

  The system notifies the user that the game is now over and prompts the user to enter their name with a pop-up. The system will then store the name and score of that player in the database.

- **Informal Scenario:**

  The player lost concentration and their spaceship collided with an asteroid.

2.3.8. System Exits the Game:

- **Current State:**

  The system state changes from pop-up menu after the player has entered their name and returns the player to the game menu.

- **Informal Scenario:**

  At this point the player will most likely be curious as to what position they placed on the high score board.

2.3.9. Player Clicks High Score Option on Menu:

- **Current State:**

  The system now displays the players position on the scoreboard.

- **Informal Scenario:**

  The player has now seen where they placed on the scoreboard and returns to the main menu.

2.3.10. Player Clicks to Quit the Game:

- **Current State:**

  The system has now been closed by the player.

- **Informal Scenario:**

  The player has stopped playing the game.

2.4 Constraints:

- **Time:**

  I must be very aware of the amount of time that I spend on each component because I must do my continuous assessment assignments during each semester and prepare for my exams. So, I will need to plan out the project, so I can have time to carry out testing.

- **Game Physics:**

  I need to ensure that the physics I implement in the game are accurate. For example, when the larger asteroid collides with smaller asteroid the velocity at which the smaller asteroid is travelling must make sense.

- **Graphics:**

  I need to ensure that the GUI is well designed because this gives the player a better experience.

- **Hardware Platform:**

  I must design a game that can be run on low-end computers.

3. Functional Requirements:

3.1 System Reliability:

- **Description** – the game must function as intended and gameplay should be smooth. There cannot be lag or screen tearing as this will affect the players experience. The game must be able to work on low-end computers with 4GB RAM with absolutely no problems.
- **Criticality** – this is very important because this will allow other components, e.g. the menu to function smoothly. It is also very critical because the game will be played on a variety of computers with different specifications, so it is essential that the game functions correctly on different types of computers.
- **Technical Issues** – the graphics implemented must be able to work with different types of computers. E.g. this will involve ensuring that the graphics in the game look good on different screen sizes, that the game runs smoothly with different CPU's and different sizes of RAM.
- **Dependencies with Other Requirements** – this will depend on the GUI and gameplay graphics.

3.2 Graphical User Interface:

- **Description** – this refers to how the user interacts with the game through graphical icons and types commands. This will involve ensuring that the graphics in the game are physically appealing, that the menu options are well designed and that player inputs are as intended, e.g. press left arrow key and the spaceship moves left and clicking on the high score option of the menu should display the high scores from the database etc.
- **Criticality** – the GUI is one of the most important requirements because a well-designed GUI always improves the players experience when they play the game.
- **Technical Issues** – I must ensure that the GUI is very well-designed and that I use the right libraries.
- **Dependencies with Other Requirements** – the design of the GUI depends on what libraries and effects I use.

3.3 Collision Detection:

- **Description** – this involves detecting collision of different sprites in the game. E.g. detecting when the large asteroids and smaller asteroids collide and detecting collision between the spaceship and asteroids.
- **Criticality** – this is critical because this is one of the fundamental aspects of the game and what will make the game difficult to beat.
- **Technical Issues** – to implement collision detection in the game it involves getting the objects to bounce off each other.

  I must ensure that when asteroids spawn on the canvas that they aren't being spawned on top of each other. This is because if two objects are overlapping they won't be able to move since our collision detection code will just assume that they are constantly colliding.

  Secondly, for each asteroid I must calculate the distance apart from themselves and every other asteroid on the canvas. I need a way to monitor collisions between each asteroid.

  When I determine how these asteroids have collided, I will need to get them to react in a realistic manner. To do so, I will have to use some functions that will give us a realistic bounce effect.
- **Dependencies with Other Requirements** – this will depend on how well the GUI is implemented and how well the algorithms that I use are implemented.

3.4 Database:

- **Description** – this is where the scores made by the player will be stored.
- **Criticality** – although it may not seem that critical, it is. Allowing players to view the high scores of other players causes them to increase the amount of time playing the game. This is just the competitive nature of gaming.
- **Technical Issues** – no real technical issues other than ensuring each players score is stored correctly and displayed correctly in the GUI.
- **Dependencies with Other Requirements** – how the high scores are displayed will depend on the GUI of the high score options on the menu.

4. System Architecture:

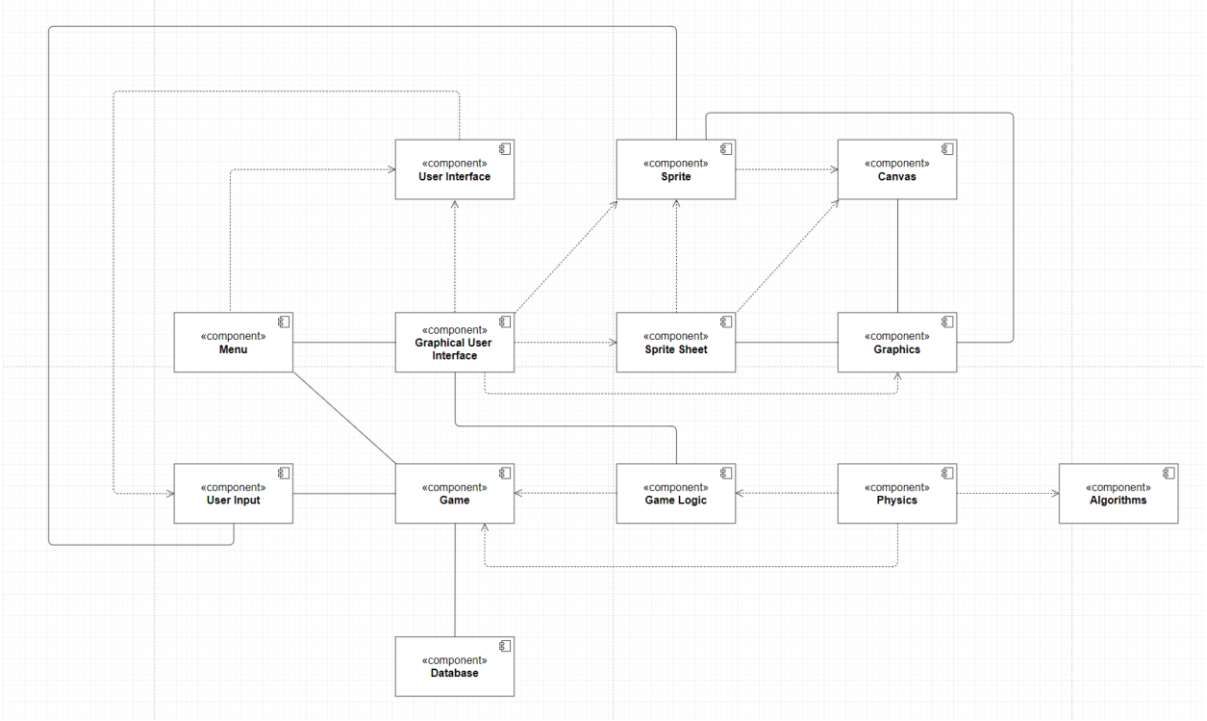


Figure 1

- The diagram above is a component diagram.
  - Dashed arrows represent dependencies.
  - A straight line represents association.

There are 13 distinct parts of the architecture:

- **User Interface** – is where interaction between the player and system occur. How the state of the interface changes will depend on the user input.

- **Graphical User Interface** – is the visual way that the user interacts with the system, e.g. icons and menus. So, the **Graphical User Interface** component will have an associative relationship with the **Menu** and **Game Logic** components. How the GUI looks will depend on how well the **User Interface**, **Graphics**, **Sprite Sheet**, and **Sprites** for the game are designed.

- **User Input** – are inputs such as clicking on buttons. The **User Input** component and **Game** component have an associative relationship.

- **Sprite** – sprites are just computer graphics on the screen that can be manipulated by the user. A **Sprite** will have an associative relationship with the **User Input** component. It will also depend on the **Canvas** component because it will be placed on the canvas.

- **Sprite Sheet** – will be composed of several sprites. How the sprite sheet looks will depend on the sprites that I use. It also has an associative relationship with the **Graphics** component. It will depend on the **Canvas** component.

- **Graphics** – this component will have an associative relationship with the **Sprite**, **Sprite Sheet**, and **Canvas** component.

- **Canvas** – will have an associative relationship with the **Graphics** component.

- **Game Logic** – the logic of the game will depend on the **Game** component.

- **Physics** – this component is responsible for how objects move and the manner they move in. How I implement the physics will depend on the **Game Logic** and the **Algorithms** I decide to use.

- **Algorithms** – Collision Detection, One-Dimensional Newtonian Formula, and any other algorithms used to develop the game.

- **Menu** – this will help the user navigate through the system. It has an associative relationship with **Graphical User Interface** and **Game** components.

- **Database** – this is responsible for storing the high scores and will have an association relationship with the **Game** component.

- **Game** – what the player will play.

5. High-Level Design:

- State Machine Diagram:

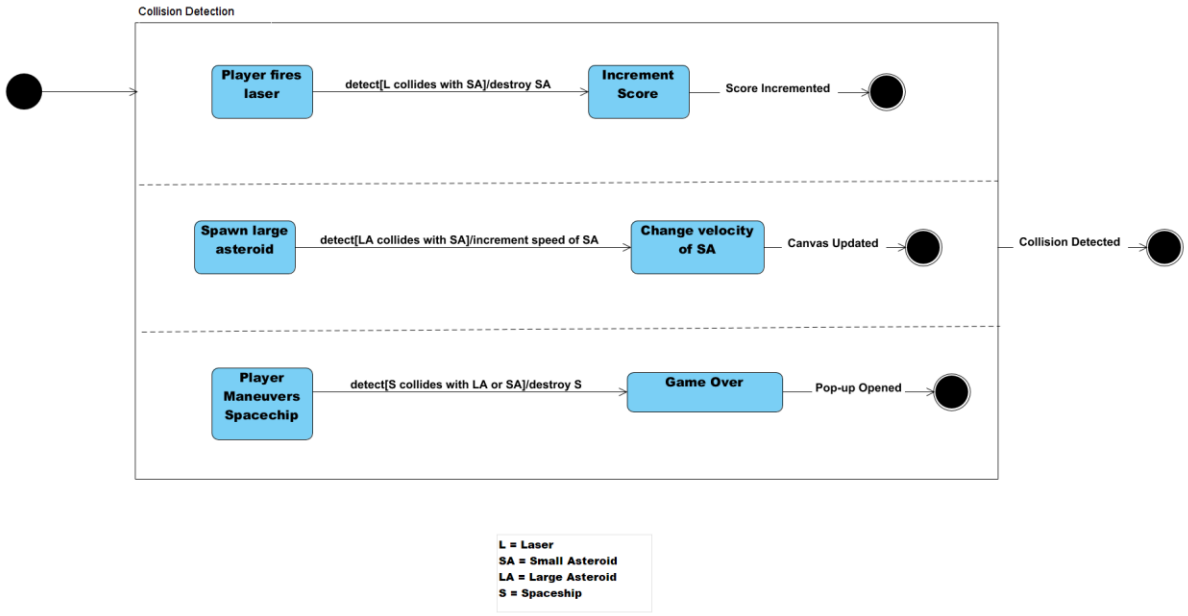


Figure 2

- Level 1 DFD:

Figure 3

Figure 3 goes through the process of what happens when the player has lost the game:

- **Request Player Name** – firstly, the game will output a pop-up which will prompt the user to enter their name.
- **Return Player Name** – that name will then be saved and sent to the database.
- **Menu Display Request** – normally when a player has finished playing, they request to go to the menu. The game will then return the user to the menu and display the options in the menu.
- **High Score Display Request** – when the player is at the menu they could choose to view the high scores. The game will then have to output the name and high score of that player and previous players.
- **Return to Game Request** – after viewing the high scores, if the player is not satisfied with their position on the scoreboard, they will most likely replay to game to get a higher score. So, the game must then display the canvas.
- **Controls Display Request** – the player can request to view the controls for the game and the game would return the details to the player.
- **Exit Game Process** – the player can choose to exit the game and the game will return a yes/no option pop-up confirming that the player wants to leave the game.

- Sequence Diagram:

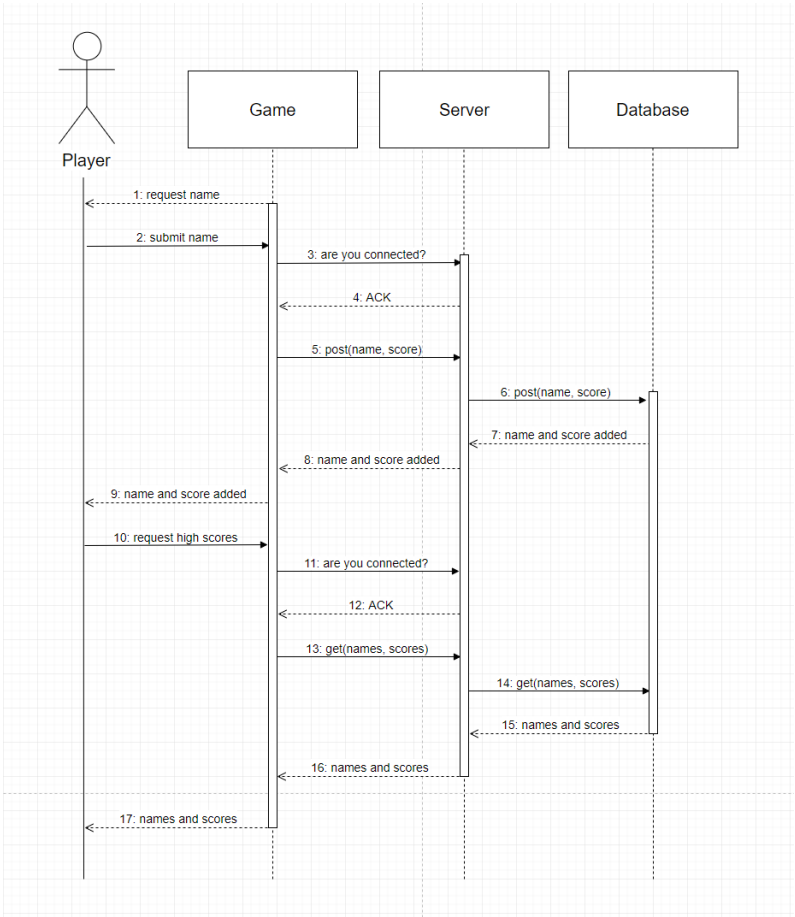
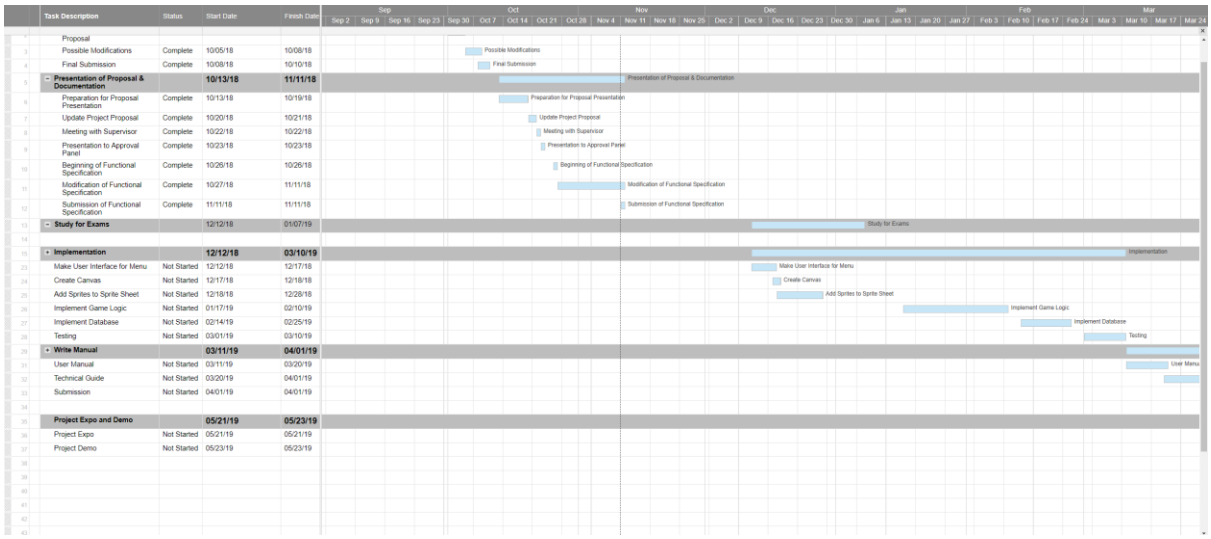
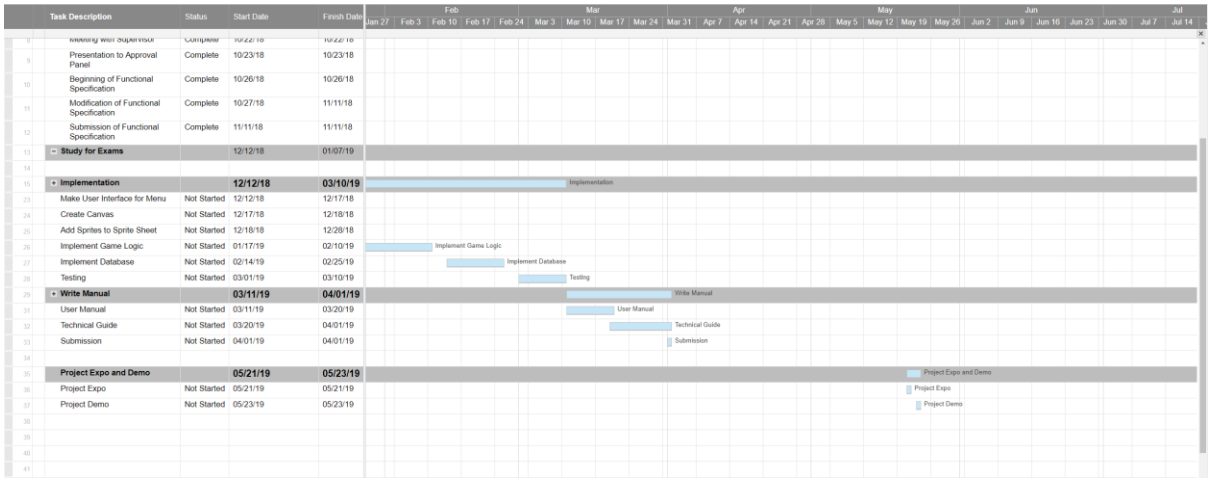


Figure 4

6. Preliminary Scheduling:



Task Description	Status	Start Date	Finish Date
Proposal			
Possible Modifications	Complete	10/05/18	10/08/18
Final Submission	Complete	10/08/18	10/10/18
Presentation of Proposal & Documentation		**10/13/18**	**11/11/18**
Preparation for Proposal Presentation	Complete	10/13/18	10/19/18
Update Project Proposal	Complete	10/20/18	10/21/18
Meeting with Supervisor	Complete	10/22/18	10/22/18
Presentation to Approval Panel	Complete	10/23/18	10/23/18
Beginning of Functional Specification	Complete	10/26/18	10/26/18
Modification of Functional Specification	Complete	10/27/18	11/11/18
Submission of Functional Specification	Complete	11/11/18	11/11/18
Study for Exams		**12/12/18**	**01/07/19**
Implementation		**12/12/18**	**03/10/19**
Make User Interface for Menu	Not Started	12/12/18	12/17/18
Create Canvas	Not Started	12/17/18	12/18/18
Add Sprites to Sprite Sheet	Not Started	12/18/18	12/28/18
Implement Game Logic	Not Started	01/17/19	02/10/19
Implement Database	Not Started	02/14/19	02/25/19
Testing	Not Started	03/01/19	03/10/19
Write Manual		**03/11/19**	**04/01/19**
User Manual	Not Started	03/11/19	03/20/19
Technical Guide	Not Started	03/20/19	04/01/19
Submission	Not Started	04/01/19	04/01/19
Project Expo and Demo		**05/21/19**	**05/23/19**
Project Expo	Not Started	05/21/19	05/21/19
Project Demo	Not Started	05/23/19	05/23/19



Task Description	Status	Start Date	Finish Date
Presentation to Approval Panel	Complete	10/23/18	10/23/18
Beginning of Functional Specification	Complete	10/26/18	10/26/18
Modification of Functional Specification	Complete	10/27/18	11/11/18
Submission of Functional Specification	Complete	11/11/18	11/11/18
Study for Exams		**12/12/18**	**01/07/19**
Implementation		**12/12/18**	**03/10/19**
Make User Interface for Menu	Not Started	12/12/18	12/17/18
Create Canvas	Not Started	12/17/18	12/18/18
Add Sprites to Sprite Sheet	Not Started	12/18/18	12/28/18
Implement Game Logic	Not Started	01/17/19	02/10/19
Implement Database	Not Started	02/14/19	02/25/19
Testing	Not Started	03/01/19	03/10/19
Write Manual		**03/11/19**	**04/01/19**
User Manual	Not Started	03/11/19	03/20/19
Technical Guide	Not Started	03/20/19	04/01/19
Submission	Not Started	04/01/19	04/01/19
Project Expo and Demo		**05/21/19**	**05/23/19**
Project Expo	Not Started	05/21/19	05/21/19
Project Demo	Not Started	05/23/19	05/23/19

7. Appendix:

Resources:

https://stackoverflow.com/

https://stackexchange.com/

https://www.google.com/

https://www.youtube.com/

https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-2017