**Table of Contents**

1. **Introduction** 3
   1. **Overview** 3
   2. **Business Context** 3
   3. **Glossary** 3
2. **General Description** 4
   1. **Product / System Functions** 4

**2.1.1**. **About**4

**2.1.2.** **Environment** 4

**2.1.3.** **Units** 4

**2.1.4.** **Offence** 4

**2.1.5.** **Defence** 5

**2.1.6.** **Turn Rules** 5

**2.1.7.** **The Core** 6

**2.1.8.** **End Game** 6

**2.1.9.** **Artificial Intelligence** 6

* 1. **Unit Specification** 7
  2. **User Characteristics and Objectives** 8
  3. **Operational Scenarios** 9

**2.4.1.**  **Novice Player Takes The Tutorial**9

**2.4.2.**  **Player Is In Trouble**9

**2.4.3.**  **Player Quits The Game Mid-Match** 10

* 1. **Constraints** 10

1. **Functional Requirements** 12
   1. **Artificial Intelligence (Opponent)** 12
   2. **Board Traversal / Path Finding**12
   3. **Animation Programming** 13
   4. **GUI Programming** 13
2. **System Architecture Diagram** 15
   1. **Game Engine** 15
   2. **Game Executable File** 15
   3. **Game Scripts** 15
   4. **Player** 16
   5. **A.I.** 16
   6. **Graphics** 16
   7. **Sound** 17
   8. **HUD** 17
   9. **Level** 17
3. **High-Level Design Diagram** 18
   1. **Run The EXE File** 18
   2. **Choose A.I. Difficulty** 18
   3. **Start Game** 19
   4. **Choose Pieces To Play** 19
   5. **Damage Enemy’s Core Health** 19
   6. **End Game** 19
4. **Preliminary Schedule** 20
   1. **Task List** 20
   2. **Gantt Chart** 21
5. **Introduction**
   1. **Overview**

The project to be developed will be a single player video game, in the form of a desktop application. The game will be a competitive turn based strategy game, based around the idea of protecting a core, and destroying the opponent’s core with an army of remote-controlled high tech vehicles at the player’s disposal.

The project’s main focus will not be game design, but rather implementing different algorithms learned by the students into a real world project. These algorithms will mainly focus on path-finding and AI.

The game will be developed using Unity 3D, a free game development engine that will be used to write code in and compile all the test versions in additional to the final game. The Unity engine will take care of the environment rendering and the compiling of the game, all other aspects of development such as gameplay programming will be done by the team from scratch.

The primary programming languages used in this project will be C# and JavaScript as they are supported by Unity 3D and have an abundance of documentation to support our development.

The game models used will be modelled in Cinema 4D. All the animations will also be created with the same software, from scratch.

* 1. **Business Context**

If the project is successful and deemed fun to play by conducting user tests, the game could possibly be deployed to Steam, an online videogame marketplace to be placed on sale.

The Steam platform allows for indie developers to create their games and submit them for user review. If deemed interesting, the users can vote to have the game placed onto the marketplace and be available to purchase and play.

* 1. **Glossary**
* **Units:** A controllable piece in the game by the player that has its own attributes and abilities.
* **Unity 3D:** A free game development engine mainly used by independent Developers.
* **Independent Developers:** Small team developers that self-publish.
* **Core:** A special game piece that once destroyed, the game ends. Like the Queen in chess.
* **Cinema 4D:** A 3D animation software for modelling the game objects as well as animating them.

1. **General Description**
   1. **Product / System Functions**
      1. **About**

The game will offer the user the challenge of defeating an AI opponent in a game based around the idea of using units with different characteristics to outsmart the player’s opponent and destroy their core located at their base.

* + 1. **Environment**

The game will take place on a hexagonal board and will reflect the style of a traditional board game you would play with your friends and family at home. A core for each player will be placed on opposite sides of the board. The board will be tile based and the movement of the player’s units will revolve around them, being limited to moving a certain amount of tiles per round depending on the parameters of the respective unit being controlled by the player. The player’s will have ninety seconds per round to make their moves, afterwards the game will move on.

* + 1. **Units**

The game will offer a range of offensive and defensive units for the player to use to defeat their enemy. It will be up to the player to decide what sort of game to play; an offensive or defensive one. These will all vary in terms of their health and attack points, mobility and special abilities as well as their creation time.

Each unit, offensive or defensive will be created by the core upon being ordered to by the player. All units will have a pre-determined TTB (time to build) value, indicating how many rounds they will take to be created.

* + 1. **Offence**

Once an offensive unit is created, they are automatically placed onto the board on a pre-determined tile and cannot be placed into storage, unlike defensive units. These units will be used to attack other enemy units and the core.

Initiating combat using an offensive unit will prevent them from moving any more tiles during that turn. Movement is based around the respective unit’s mobility parameter. Note that you are not required to move during a turn if you wish to do so.

If a player’s unit attacks an enemy unit, they will deal the damage equivalent of their respective attack point parameter, taking away that amount of health points from the enemy unit. Regardless of whether or not that destroys the enemy, the enemy will also inflict damage onto the attacking unit, equivalent of their own attack points.

Units cannot be repaired, and once destroyed vanish from the board. You cannot attack your own units.

* + 1. **Defence**

You will be able to place defences within **X** (to be determined through play-testing) tiles, relative to your core. Additionally, defensive units will be able to be placed within **Y** tiles of your offensive units. A max of three defensive units can be placed into storage at a time. Defensive units will be created by the same core creating offensive units, forcing the player to pick a priority, offensive or defence, adding an additional layer of complexity and strategy to the game.

The defensive units will consist of walls, turrets and traps. Each of these (except traps) can be destroyed by anyone, regardless of whether or not they placed the unit themselves.

Trap vehicles will be able to traverse the board and place hidden traps for the enemy. A trap will be triggered when a tile containing a trap is entered by a unit. All traps are dangerous to all players, as you are not exempt from triggering traps which are placed by you. Once you place a trap, you must move away from the tile immediately or else the trap will be triggered next turn.

Traps will be used to inflict damage or slow down your opponents and will consist of a few different types, such as walls, EMPs and bombs.

* + 1. **Turn Rules**

During a single turn, a player will be able to:

1. Move all their units currently placed on the board.
2. Place any defensive units currently in storage.

**2.** Start the creation of a unit.

**3.** Attack the opponent’s units or core.

**4.**  Place traps.

**5.** Collect loot.

**6.** Do nothing.

Each turn is limited to ninety seconds max, and can be completed earlier by the press of a button by the player who’s turn it is.

**2.1.7. The Core**

The core is the most important part of the game. It is responsible for creating units and acting as the goal for each player.

A core will have health points, but will not react to damage the same way as normal units will.

The core will:

1. Have three health points.
2. Be immobile.
3. Create offensive units.
4. Create defensive units.
5. Only be attacked by one unit per turn.
6. Only lose one health point per round.
7. Destroy any unit regardless of their health points once attacked.
8. End the game if destroyed.

**2.1.8. End Game**

Once a core is destroyed, the game ends, winning the game for the player whose core is still intact.

A time limit will be determined by the player before starting a match for the whole game. Once the time runs out, the player whose core has less health points will automatically lose. If the health points of each core are the exact same, the game will go into overtime for fifteen minutes. The next player to inflict damage onto the enemy core will win the game, regardless of whether or not that damage is enough to destroy the core. If nobody inflicts any damage to any core within the fifteen minutes, the game will end and be considered a draw.

**2.1.9. Artificial Intelligence (Opponent)**

The enemy AI will be based around state machines, switching between them depending on what situation it finds itself in. For example, if a unit with low health is near, follow and attack them, or if their core has little health, play more defensively.

The game will offer two difficulty levels, easy and hard.

The easy difficulty will offer a small challenge to the player, ideal for those getting familiar with the game. The AI will focus mainly on attacking you, utilising little defensive tactics.

The hard difficulty will offer a greater challenge to the player, by both being offensive and defensive, utilising all aspects of the game.

* 1. **Unit Specifications**

**Vehicle Unit Specification**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vehicle  Unit | Health Points | Attack  Points | Mobility | Range | Time To Build |
| Car | **2** | **2** | **2** | **2** | **2** |
| Bike | **1** | **1** | **4** | **1** | **3** |
| Tank | **5** | **3** | **1** | **2** | **5** |
| Drone | **2** | **1** | **3** | **4** | **4** |
| Trap Car | **2** | **0** | **2** | **2** | **3** |

***Note:*** *All unit attributes are subject to change for game balancing at the project testing stage*.

**Health Points:** These are lost when attacked by an enemy unit or damaged by a trap and represent the life of a vehicle. Once the health point value reaches zero or below, the unit is destroyed and vanishes from the board.

**Attack Points:** When attacking an enemy unit, this is the amount of health points that you will take away from the unit being attacked.

**Mobility:** This is the amount of tiles a unit can traverse in one round. Note, they are not required to use all of their mobility points if they choose to do so.

**Range:** This is the range of tiles from within you can deal damage to an enemy unit or core.

**Time to Build:** The amount of turns it takes for the unit to be created by the core. Only one unit can be built at a time.

**Trap Vehicle Specification**

|  |  |
| --- | --- |
| Trap | Effect |
| Sudden Wall | **Builds a three tile length wall** |
| EMP | **Disable the vehicle that triggered the trap for two rounds** |
| Bomb | **Deal three damage to the vehicle that triggered the trap, and one damage to any surrounding units** |

***Note:*** *All unit attributes are subject to change for game balancing at the project testing stage*.

**Sudden Wall:** Once a vehicle attempts to enter a tile that contains this trap, they will be stopped and three tiles side by side in front of said vehicle will be replaced by walls. The unit will not be able to move again that turn.

**EMP:** Upon entering a tile that contains this trap, the unit will be ‘shut down’ for three rounds, rendering it immobile and in-able to attack for the next two rounds.

**Bomb:** When this trap is triggered, any vehicle on the tile will lose three health points. Additionally, any vehicles on all adjacent tiles will lose one health point.

**Defensive Unit Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| Defensive Unit | Health Points | Time To Build | Description |
| Basic Wall | **3** | **2** | **Takes up one tile** |
| Long Wall | **4** | **5** | **Takes up three tiles** |
| Turret | **3** | **6** | **Deals two damage with a range of three tiles** |

***Note:*** *All unit attributes are subject to change for game balancing at the project testing stage*.

**Basic Wall:** A wall that can be placed to block paths.

**Long Wall:** A more efficient wall, but takes longer to create.

**Turret:** An automated turret that attacks enemy units at random. Max of three placed per player at a time.

* 1. **User Characteristics and Objectives**

The ideal audience for the project will be young adults that enjoy strategy videogames or board games and are somewhat familiar with this genre of games.

The users should be:

1. Intermediate or above in terms of skill at computer usage.
2. Regular consumers of videogame entertainment.
3. Be familiar with the turn based genre of videogames.
4. Problem solvers.
5. Strategists.
6. Patient.
7. Competitive.

We expect the users to be patient strategists who enjoy taking their time to come up with different ways of challenging their opponents. The game will be of a slower pace than most titles currently offered and played by the mainstream community of videogame consumers. This market does exist and is being targeted directly towards them, without taking concessions to appeal to a wider audience.

However, accessibility is important so an easier difficulty is offered to anyone who may not be too familiar with turn based strategy games and are looking to get into them, but for the full experience they will need to advance to a higher difficulty at some point.

At the highest level, the game will attempt to offer the user hours of enjoyable gameplay, by challenging their strategic and problem solving skills, helping them to grow in addition to just generally have a good time doing so.

* 1. **Operational Scenarios**
     1. **Novice player takes the tutorial.**

Although not the target market we are going for, the game will offer an easy difficulty in case a user does not have much or any experience in turn based strategy games. The following events are all in-order of their occurrence.

1. The user will from the main menu selecting the tutorial and be placed into a game.
2. They will be placed onto the board and start to familiarize themselves with the user interface as the game guides the player through it, highlighting different parts and offering an explanation.
3. They will be asked to create a unit and will do so by selecting the core and the highlighted unit the game will force you to choose.
4. After its created, they will use it to move through the board and destroy a pre-determined enemy unit.
5. They will then be asked to place a defensive structure to block an enemy from attacking them.
6. Finally, a time jump will happen and the player will be asked to attack the enemy core, destroying it and winning him the game.
7. The player will then be returned to the main screen.
   * 1. **Player is in trouble.**

The player is losing the game, and has been forced to retreat to their base to protect their core and offer defensive support to the core. The enemy’s units have surrounded the base.

1. The player places three walls from their storage, completing a wall that fully encloses the base and no ground units can enter nor leave the base.
2. The core has one health point left.
3. They begin to create a tank unit.
4. They end their turn.
5. The enemy ground units attack the walls to destroy them.
6. The player turrets that have been previously placed are attacking the enemy ground units.
7. A wall is destroyed and the enemy drives in, entering a trap and being destroyed.
8. The enemy ends their turn.
9. The player unit sends their ground units through the small opening to attack enemy units and destroy two, but also destroy themselves in the process.
10. The core is unprotected.
11. The player ends their turn.
12. The enemy drives in and attacks the core for the last time, defeating the player.
13. The game ends.
    * 1. **Player quits the game mid-match.**

The player is losing the game, and is getting frustrated.

1. The player’s core has one health and the player is hot-headed.
2. They decide to quite the game and press the quit game button.
3. A message box pops up stating that, ‘Quitting the game will result in the loss of progress, are you sure you want to quit?’.
4. A choice of two buttons is offered, ‘Quit’ and ‘No’.
5. The player presses the ‘Quit’.
6. The game ends and progress is lost.
   1. **Constraints**
7. The game will only be available to be played on a Window’s machine as the project team does not have access to a MacOS hosted machine, which would allow for development for their preferred platform, the iPhone.
8. The game will be single player only, as in addition to time constraints, the group was made aware that the project must focus more on the implementation rather than game design. This makes developing a two player mode to be too little of an award for a lot work.
9. Artificial Intelligence capabilities. The project team are interested in AI and would like to learn more about it and have so chose to do a project that focuses heavily on AI, despite having little knowledge on the inner workings of it in real world applications. The team will attempt to create as complex of an AI as possible for the harder difficulty mode, but are aware of their own lack of knowledge and the possibility of struggling to get it to a satisfactory level of complexity so concessions may have to be made depending on progress being made throughout the development.
10. Development will take place largely in Unity 3D, a platform that neither partner has much experience in, meaning that some time will have to be taken to learn how to use it properly first before the game can be properly developed, possibly making the team to lose valuable development time and forcing them into crunch time towards the project due date.
11. **Functional Requirements**
    1. **Artificial Intelligence (Opponent)**

**Description:**

The game will not support two or more human players, so creating a functioning AI is essential. The AI will have two difficulties, easy and hard. It must act intelligently, and even on the easiest difficulty offer at least a little challenge to any player. Depending on what situation it finds itself in the game, it must make smart decisions in order to give it the best chance of winning the game and being fun to play against.

**Criticality:**

This is the most important part of the entire game. No matter how polished all the other features are, if the opponent offers no challenge then the rest of the game is rendered useless and not fun.

**Technical Issues:**

Building an intelligent AI system can be difficult, and as neither of the partners for this project have much experience with AI in the real world, it can prove to be difficult to create an enemy that acts as human as possible to offer a challenge to the player, especially on the harder difficulty.

**Dependencies with other requirements:**

This is the driving force behind the game being fun and all other features depend on the AI to work for the game to be enjoyable. It will depend heavily on state machine AI algorithms.

* 1. **Board Traversal / Path Finding**

**Description:**

For the units to traverse the board, certain algorithms will be needed to calculate the shortest paths to certain tiles and spaces that are blocked, occupied, etc.

**Criticality:**

Correct traversal is very important to make sure the gameplay is fair and balanced.

**Technical Issues:**

Incorrectly implemented algorithms can lead to bugs such as multiple units entering the same tile, going through walls, etc. Little experience currently with applying algorithms such as Dijkstra’s to real world applications, may take time to learn.

**Dependencies with other requirements:**

All units will depend on these algorithms working correctly to provide them the shortest path to reach their desired tiles, avoid walking through walls or enemy units.

* 1. **Animation Programming**

**Description:**

These days all games need appealing visuals, and this game is no exception. It will be important to make sure that correct animations play during the correct events, such as a vehicle taking the correct path to a given tile, e.g. Going from tile A to tile D through B and C instead of skipping.

**Criticality:**

Not critical, but important to make the game appealing and inviting.

**Technical Issues:**

Can lead to clipping (moving through objects) if not implemented correctly, or playing a death animation when placing a trap. Compromises to the quality of animation may also have to be made if short on time.

**Dependencies with other requirements:**

Will be dependent on other algorithms and states to provide the correct requests for animations to be played. Will not affect other features.

* 1. **GUI Programming:**

**Description:**

The general user interface must display the correct information to player. This includes everything from the main menu, the core’s and unit’s health points, correct statistics for each unit and more.

**Criticality:**

Can lead to a confusing and unpleasant gaming experience if the wrong information is being displayed, leading to frustration and perhaps forcing the player to quit the game all together.

**Technical Issues:**

With lots of information to display, not only is it important to make sure the correct data is being shown in the correct places, but the design of the UI is vital as well. It is easy to confuse a player and lead them to frustration.

**Dependencies with other requirements:**

Will be dependent on the data being stored by the game to show the correct information for the right players and units.

# **4. System Architecture Diagram**

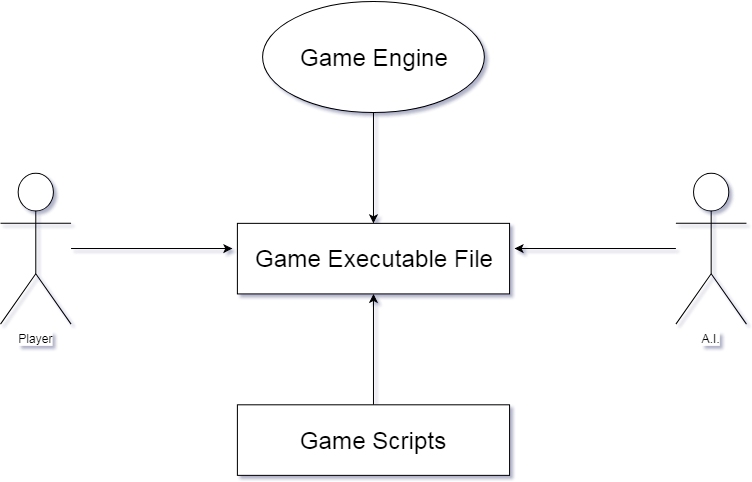


Figure 1

## **4.1. Game Engine**

The game will run on the Unity 3D engine. This will be the front end of the game where the game board and the pieces will be presented. The game engine will handle the graphical side of the game as well as operate the audio present in the game.

## **4.2. Game Executable File**

The game will be a single player game run by an executable file. This will be a regular application file that once run will launch the game.

## **4.3. Game Scripts**

The scripts that will run the game will be written in C# or JavaScript. This will be the back end that will make the game work as intended. A piece making a move or a piece going into construction are examples of scripts that will have to be written.

## **4.4. Player**

This will be a single player game; hence the player will be playing against an AI player.

## **4.5. A.I.**

The A.I. will have two difficulties: A hard and an easy mode which the user will be able to choose.

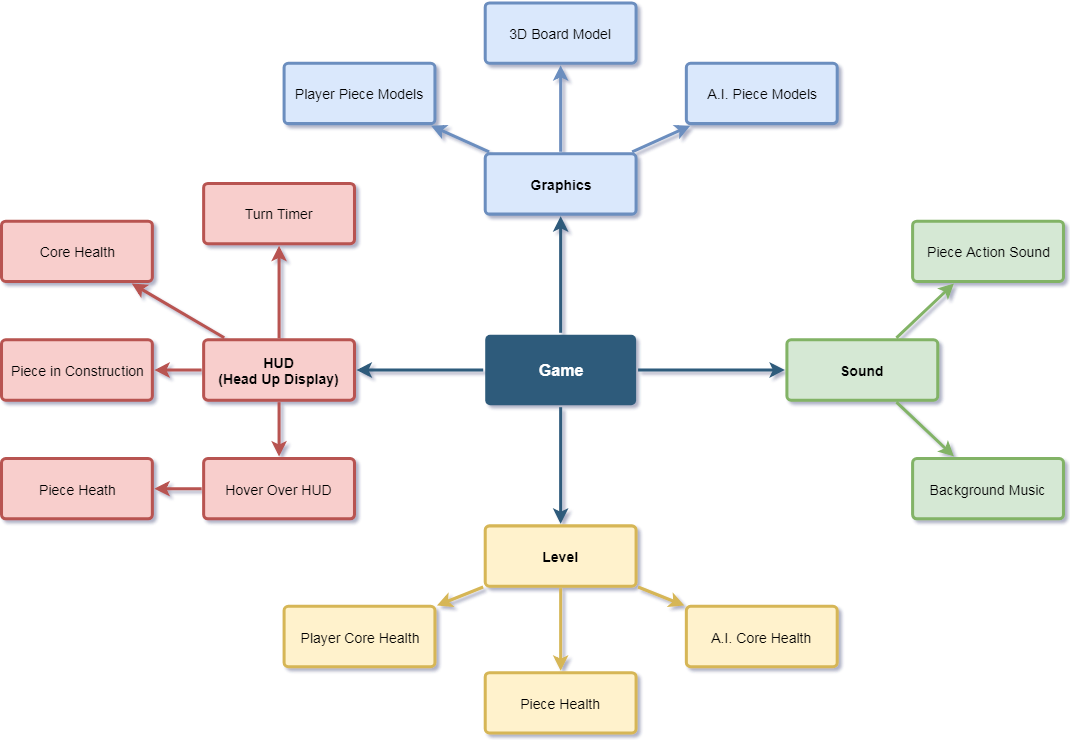


Figure 2

## **4.6. Graphics**

The graphics will be based on the modelling part of the game, such as the Board model as well as the models for each of the pieces.

## **4.7. Sound**

Each piece will have their own action sound that will be played once an action takes place i.e. an offensive piece makes an attack and a defensive piece interacts with an enemy piece.

## **4.8. HUD**

The Head Up Display will show the time left for the turn to end, the health of the player’s core and any pieces that may be currently in construction in the core. Each piece on the board will have their own hover over head up display that will include the piece’s attributes.

## **4.9. Level**

The level will be the actual game. It will have two major variables: The player’s core health and the enemy’s core health. It will also have variables for each of the piece’s health.

# **5. High-Level Design Diagram**

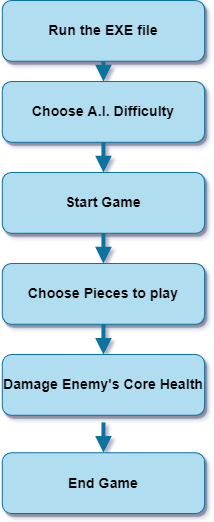


Figure 3

## **5.1. Run the EXE file**

The user will run the executable file which will launch the game.

## **5.2. Choose A.I. Difficulty**

The user will be able to choose between two difficulties. An easier one which will be more basic and beginner friendly, and another more difficult one more suitable for more experienced players who are looking for a real challenge.

## **5.3. Start Game**

Once the game starts, the user will pick their first piece and it will go into construction. After the construction period is over, the piece will be placed on the board.

## **5.4. Choose pieces to play**

As the game goes on, the user will be able to create more pieces which after a certain amount of turns will be placed on the board and will be used either offensively or defensively.

## **5.5. Damage Enemy’s Core Health**

The objective of the game is to destroy the enemy’s core. By using the offensive pieces, the user will be able to lower the enemy’s core’s health and once the core gets destroyed the user wins.

## **5.6. End Game**

Once the game is over, the user will be brought into a winning or losing screen with the amount of turns played and time taken. From there, the user will be able to either quit the game or play again.

# **6. Preliminary Schedule**

## **6.1. Task List**

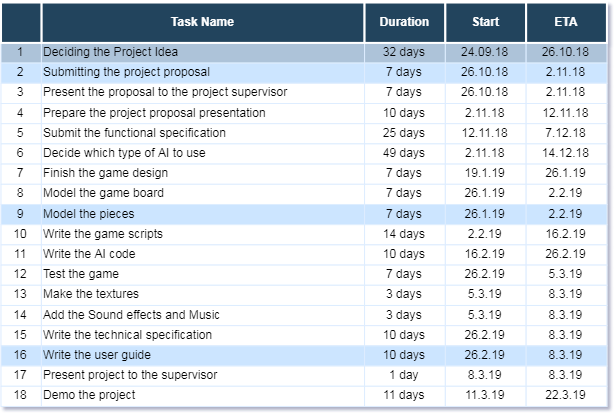
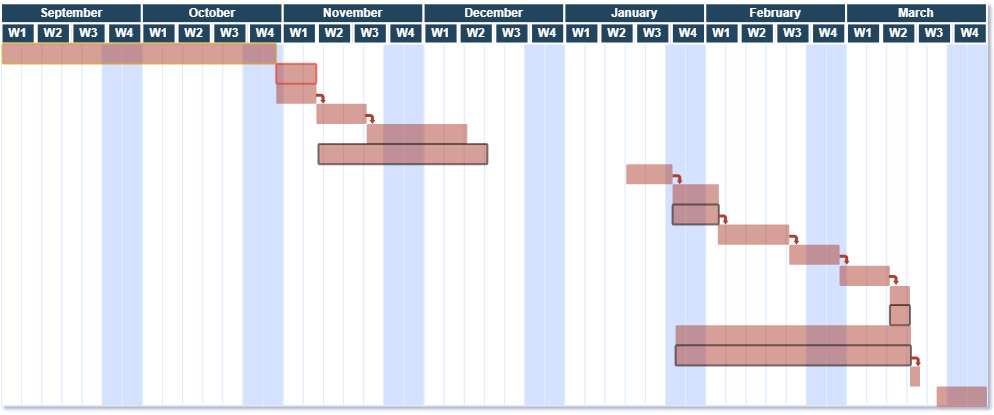


Figure 4

This table shows all the planned tasks and the deadline that we plan on hitting throughout the duration of the completion of this project. Each task as a start date in which we plan on starting the specified task as instructed. Each task also has a preliminary estimated time of completion and its preliminary duration. The estimated time of completion is the date that the team expects the task to be fully finished. Every deadline is a rough estimate, therefore, the dates may vary.

## **6.2. Gantt Chart**



Figure

The Gantt Chart shows how the team planned the tasks’ deadlines throughout the year. The year is split throughout the months, and the months split within 4 weeks each.