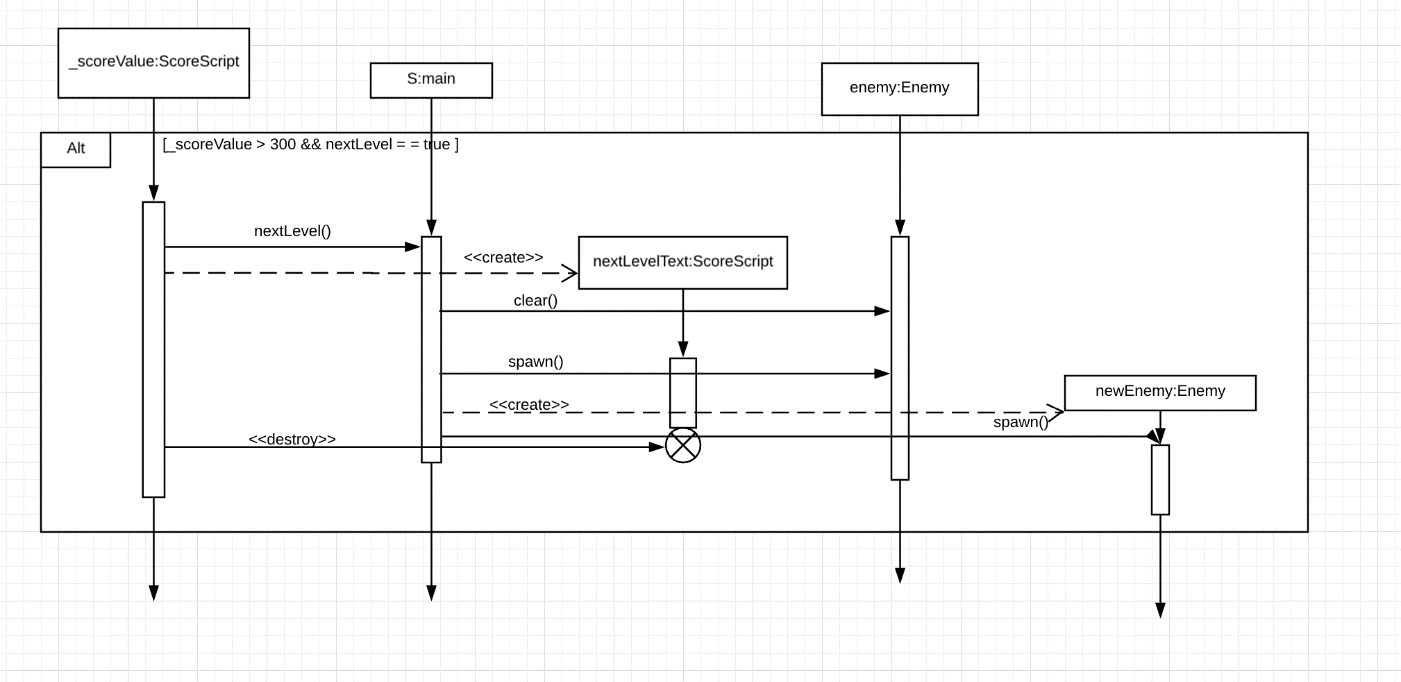
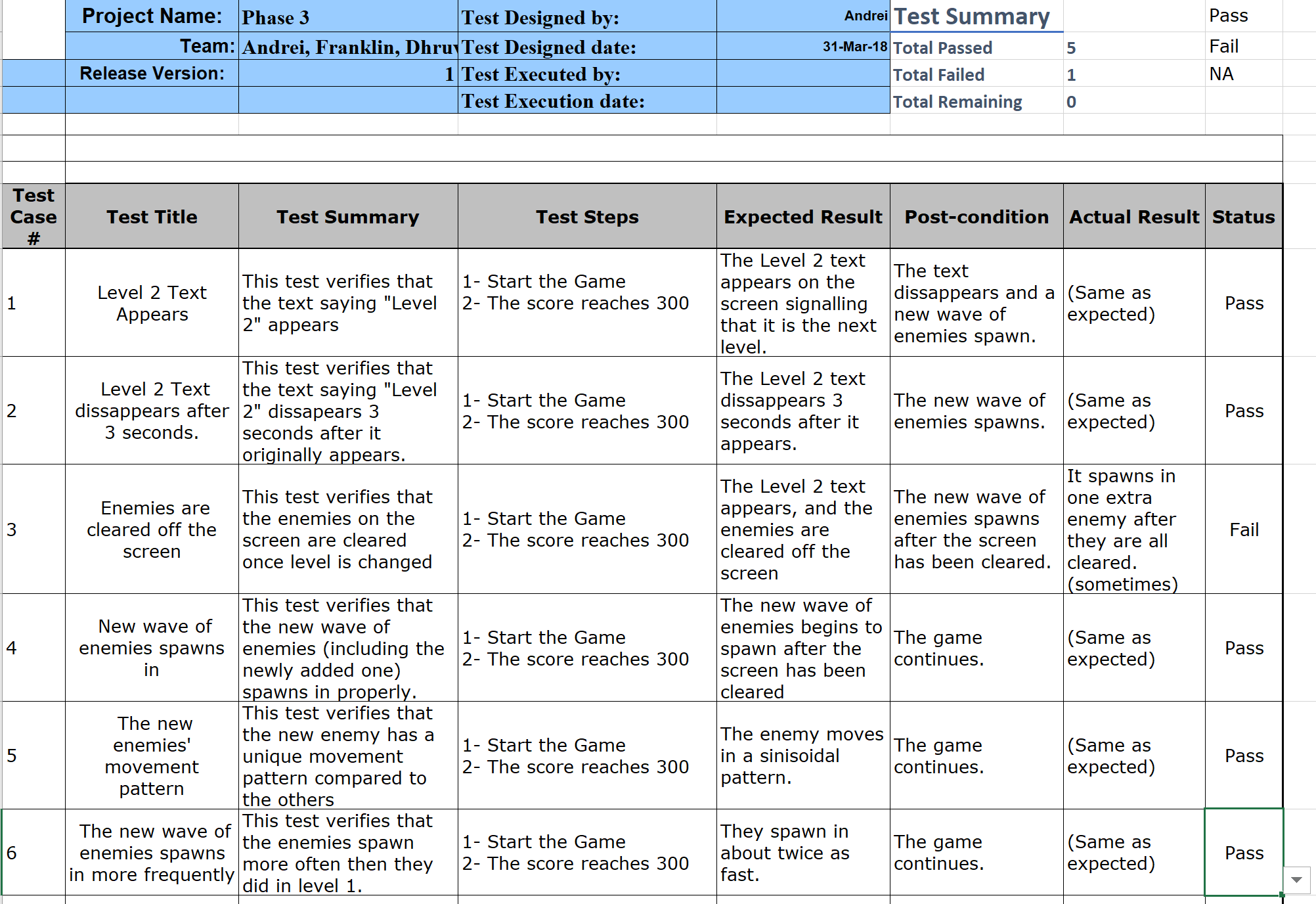
SE 2250 Project Phase 3

Andrei Popescu

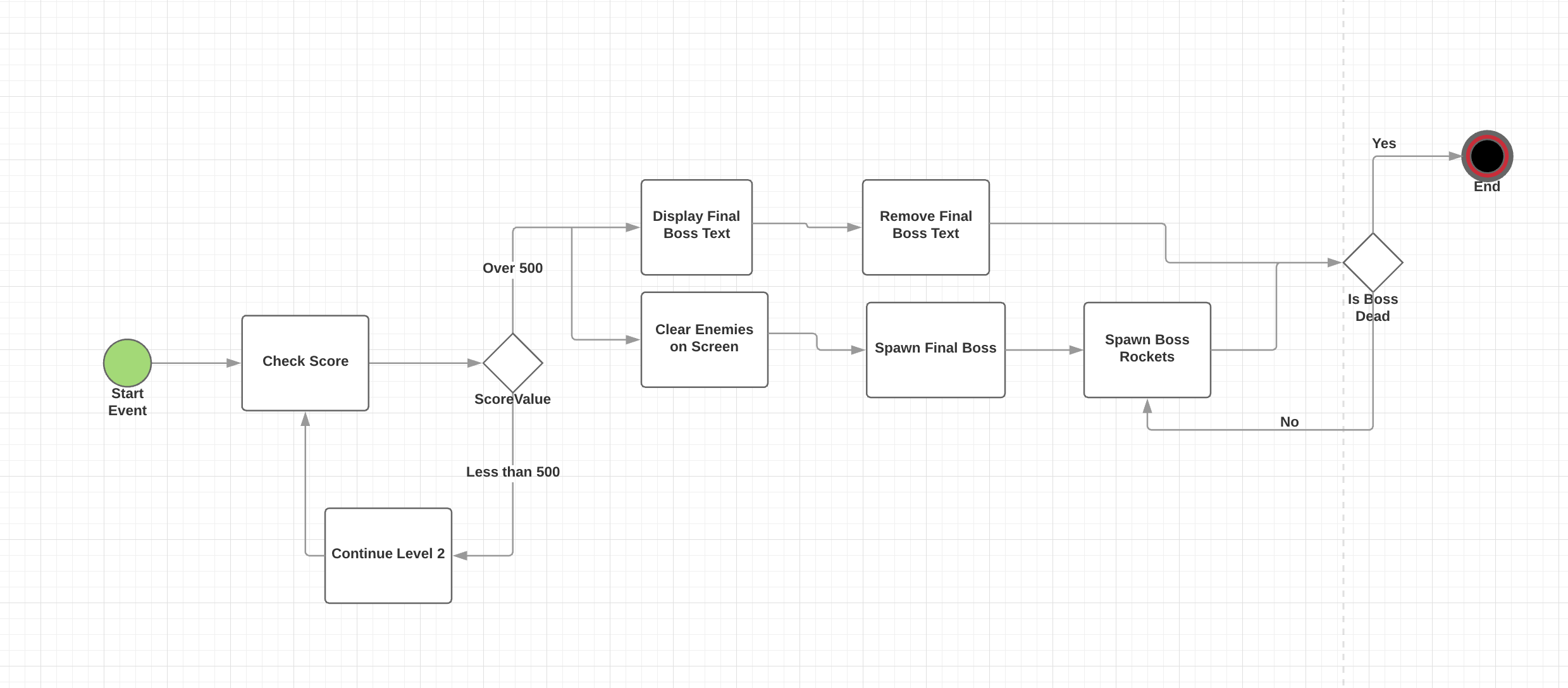
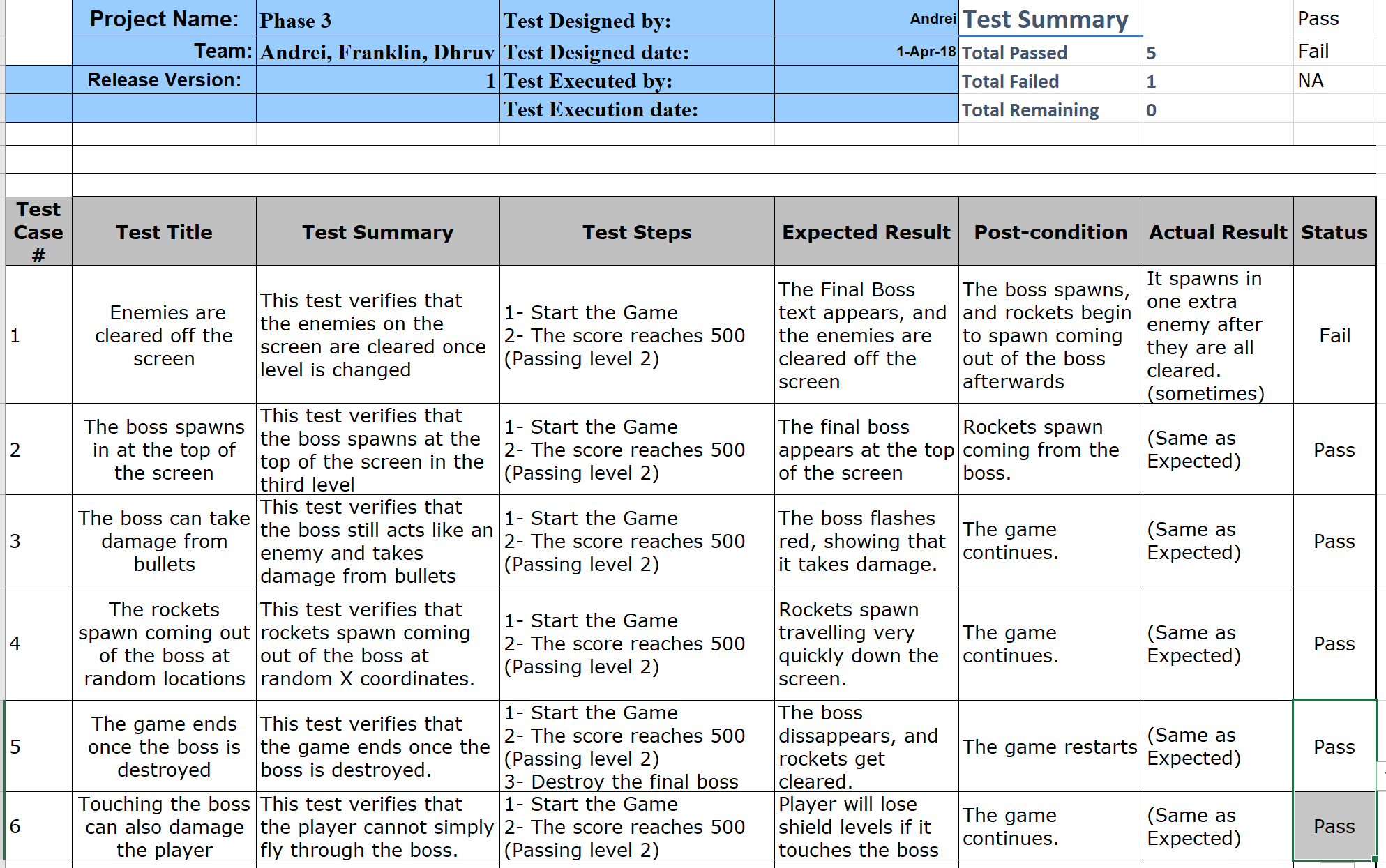
Franklin Liu

Dhruv Tayl

**Software Feature 1:** Level 2

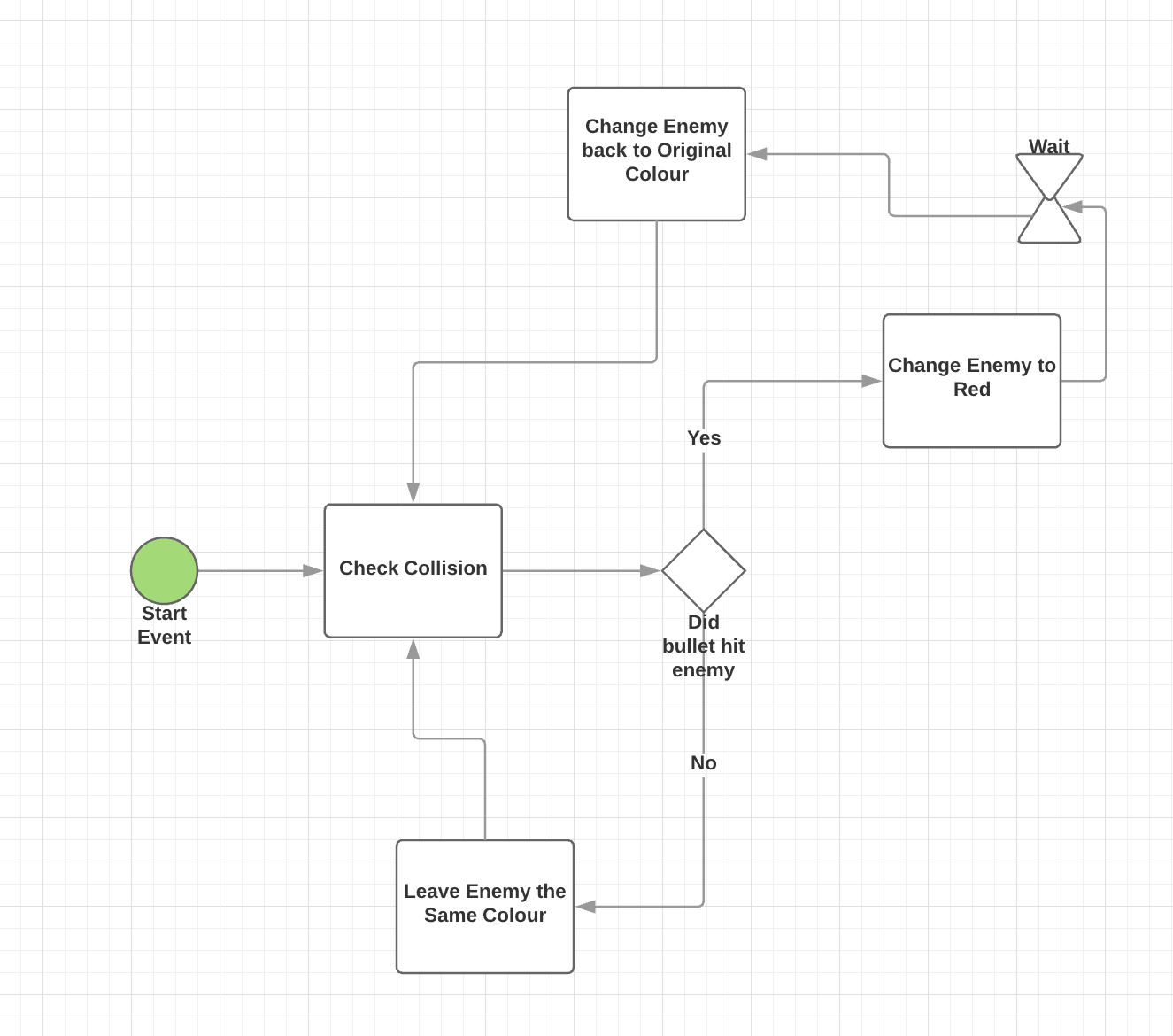
1. Level 2 will be initiated after a certain amount of score is achieved. It will have a new wave of enemies that spawn more frequently and also move slightly faster. It will also include a new type of enemy that moves along a sinusoidal path and moves much quicker then the other enemies. Since this new enemy moves a lot quicker and more erratically, it becomes harder to hit, therefore it will have a reduced amount of health compared to the other enemies.
2. The score is stored in a private variable within ScoreScript, so this variable will be used to determine whether if it is time to move on to the next level or not. Once the score reaches a certain value it will trigger an “if” statement within the update function where it uses the public static variable “S” to call the nextLevel() function within main. Also, within the “if” statement, making the “LEVEL 2” text appear and disappear is handled using invoke, calling a function that makes the text disappear after the program waits 2 seconds. Within the nextLevel() function, the level counter variable is increased, all enemies on the screen are cleared, and the spawnWave2() function is called. spawnWave2() spawns all the enemies from level 1 quicker, and also spawns a new type of enemy that follows a different movement path. The new enemies’ movement path is handled within Move() in the enemy class.
3. Sequence Diagram:
4. Test Plan:

**Software Feature 2:** Final Boss

1. This will be incorporated after level 2, once a certain score threshold has been achieved. The level will then switch and a final boss will appear that has a substantial amount of health and frequently fires rapid rockets, forcing the player to rely on reflexes to defeat it. After the final boss has been defeated, a “You Win” screen will pop up, and the game will then restart.
2. Once again this uses the \_scoreValue variable in ScoreScript to determine whether if it is time to move on to the next level. Once it reaches a certain score, it will call upon an “if” statement in the update() of ScoreScript, where it calls the finalBoss() function in the main after displaying and removing the “FINAL BOSS” text. In finalBoss() it will spawn the boss itself at the top of the screen, and then call a method rockets() to spawn its rockets over and over until the boss had been defeated. Once the boss has been defeated, it will give a surplus of score to the player, which can be detected by \_scoreValue and used to restart the game using DelayedRestart() in main.
3. Activity Diagram:
4. Test Plan:

**Software Feature 3:** Enemies turn red when hit by bullets

1. This is incorporated to all enemies, including the final boss. It is just so that the player has a visual representation of damage being dealt to the enemies, so they don’t have to guess if their bullets hit or not. It also makes the game look more polished as well.
2. This mainly takes place in the Enemy class, however it uses the utils class to get access to all the materials with getAllMaterials(). In the awake() function in enemy, an array of colours is created from the materials acquired from getAllMaterials. Within the update() function, it will constantly call the unShowDamage() function every certain amount of seconds, making the enemies appear to be in their normal state. However, in OnCollisionEnter(), there is a call to showDamage() where it colours the enemy red for a split second until unShowDamage is called again in update().
3. Activity Diagram:



1. Test Plan:

