Name: Shruti Debnath Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

[**Instructions**: Remove everything that is not a heading below and fill in with your own diagrams, etc.]

## Brief introduction \_\_/3

The feature I am working on is updating the players over all strength boards where I will implement functions to update the score board, health bar and power ups.

After entering the battlefield, my function will initialise the score, health bar and power ups which will change throughout the game. The score depends on the coins the players collect and the coins will be visible so it will depend on the player who can grab the coins as fast as possible. The health bar will be initialised as hundred which is the highest health bar each player can have and it will decrease throughout the game depending on how the player gets hitted. There will be treasure which can affect their health bar. For the power ups, each player will have special power they will be able to use in their game but that will be just for a certain amount of time.

Therefore, I am responsible for keeping track of a player's Score, Power ups time and Health Bar throughout the game.

## Use case diagram with scenario \_\_14

[Use the lecture notes in class.

Ensure you have at least one exception case, and that the <<extend>> matches up with the Exceptions in your scenario, and the Exception step matches your Basic Sequence step.

Also include an <<include>> that is a suitable candidate for dynamic binding]

Example:

### Use Case Diagrams



### Scenarios

**Score Board**

**Name:** Coins Collected

**Summary:** The player collected the coins that were visible on the battlefield.

**Actors:** The player, Opponents, Coins.

**Preconditions:** Players entered the battlefield and the game has begun already, Coins are visible suddenly.

**Basic sequence:**

**Step 1:** The players went to grab the coins.

**Step 2:** Moving towards the coins as fast as possible by using moves.

**Step 3.1:** Checks for the opponent's move.

**Step 3.2:** Both try to get coins and engage in tricking each other.   
**Step 4:** Whoever collects the coins, score adds up to his profile.

**Exceptions:**

**Step 1.1:** Get killed while attacking each other to get the coins.

**Step 1.2:** The one who is alive is the winner also gets the score.

**Step 1.3:** Both of them got killed, it will be a game over.

**Step 2:** Both players missed coins, no score added up to anyone’s profile.

**Post conditions:** The score that has been added to which player’s profile will be displayed and also the game final scene (if the game is over).

**Priority:** 2\*

**ID:** C01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Health Bar**

**Name:** health\_count

**Summary:** The health\_count will change the value of health\_bar for the players. If a player finds treasures that boost up his health bar then the health bar will increase while if the player finds a treasure that lowers his health condition then it will change his health bar. Also, the health bar will change based on where the player gets hitted by the opponent such as if the player gets hitted on head it will lower the health bar by 5 while in case of hand injury it will lower by 2.

**Actors:** Opponent, Player.

**Preconditions:** Health Bar has been initialized. Treasures has been placed.

**Basic sequence:**

**Step 1.1:** Hitted by the bot or the opponent.

**Step 1.2:** Physical part that has been injured.

**Step 2.1:** Find treasure that boosts the health bar.

**Step 2.2:** Find treasure that lowers the health bar.

**Step 3.1:** Change of health bar.

**Step 3.2:** Health bar is zero, game over.

**Step 4:** Calculate and show the result on the screen.

**Exceptions:**

**Step 1:** Finding treasure that boosts up the health bar while the health bar is already highest.

**Step 2:** No change in health bar.

**Post conditions:** Calculated value is displayed.

**Priority:** 1\*

**ID:** C01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Power Ups**

**Name:** power\_timer

**Summary:** The power\_timer will keep track of times that how many times the player used his special power and for how many seconds. There will be restrictions for timing. power\_timer will check how many times has remained to use such as the player can not use his special power more than 5 times in the game and each time he can use just for 5 seconds.

**Actors:** Player, Scene.

**Preconditions:** Game has started, power ups are initialized.

**Basic sequence:**

**Step 1:** Need of using the power

**Step 2.1:** Count power\_ups

**Step 2.2:** If there are unused powers, then let the player use

**Step 2.3:** If no, then show it on the screen.

**Step 3.1:** When using the power, start the timer to keep track of time

**Step 3.2:** After 5 sec, stop the timer and take away the power from the player.

**Step 4:** Change the power\_ups count

**Exceptions:**

**Step 1:** Trying to use it while there are no unused powers.

**Step 2:** Maybe use the coins to buy recharge powers.

**Step 3:** Show on screen, no power left to use.

**Post conditions:** Change the power\_ups count on the player profile.

**Priority:** 2\*

**ID:** C01

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

[Get the Level 0 from your team. Highlight the path to your feature]

Example:

### Data Flow Diagrams



### Process Descriptions

Score -

1. When the coins appear in the game the players will try to grab them if they want and also they can attack each other to take the coins.
2. Whoever gets the coins first will have the score added to their profile.
3. During the attack the health bar will be affected as well.

Health Bar -

1. Health will be initialized as hundred.
2. Throughout the game the health will be changed.
3. Depending on the injured spot, health will decrease.
4. There will be hidden treasure in the game if the player can find that depending on the treasure their health will change. Some treasure can be poisonous and lower their health bar and some can be beneficial that will increase their health bar.
5. Health bars can not go over 100 and it is essential to decide who will win.

## Acceptance Tests \_\_\_\_\_\_\_\_9

[Describe the inputs and outputs of the tests you will run. Ensure you cover all the boundary cases.]

**Goal :** Ensure the player's profile has been updated throughout the game depending on their moves.

**Steps :**

1. Launch the game.
2. Get hitted or find treasures then change on health bar
3. Coins appear, try to collect them, if collision happens then health bar changes
4. Try to use the special power and check how much has been used, keep track of the time for each use.

**Expected Results :**

1. Game is ongoing
2. Change on Health
3. Change on Score
4. Power ups updated.

| Tests ID | Test Description | Steps | Expected Results |
| --- | --- | --- | --- |
| T01 | Score | Go collect coins.  Attack opponents to get coins.  Get coins or not. | 1. Update score based on whether the player collected the coins or not |
| T02 | Health Bar | Attacking any body part  Attacking head  Finding good Treasure  Finding poisonous treasure. | 1. If the player gets attacked on any body part the health bar will lower by 3. 2. If the player gets an attack on the head, the health bar will lower by 5. 3. If the player grabs beneficial treasure, it will raise his health bar by 5. 4. If the player grabs poisonous treasures the health bar will lower by 8. 5. Health bar can not exceed 100. 6. When the health bar is 0, it means game over and the player loses. |
| T03 | Power ups | Select power to use it | 1. Check for remaining powers (unused). 2. If any powers are left to use, then let the player use, otherwise show that no power remains. 3. If the player is using the power, keep track of the time, then take the power back after 5 sec. |

## Timeline \_\_\_\_\_\_\_\_\_/10

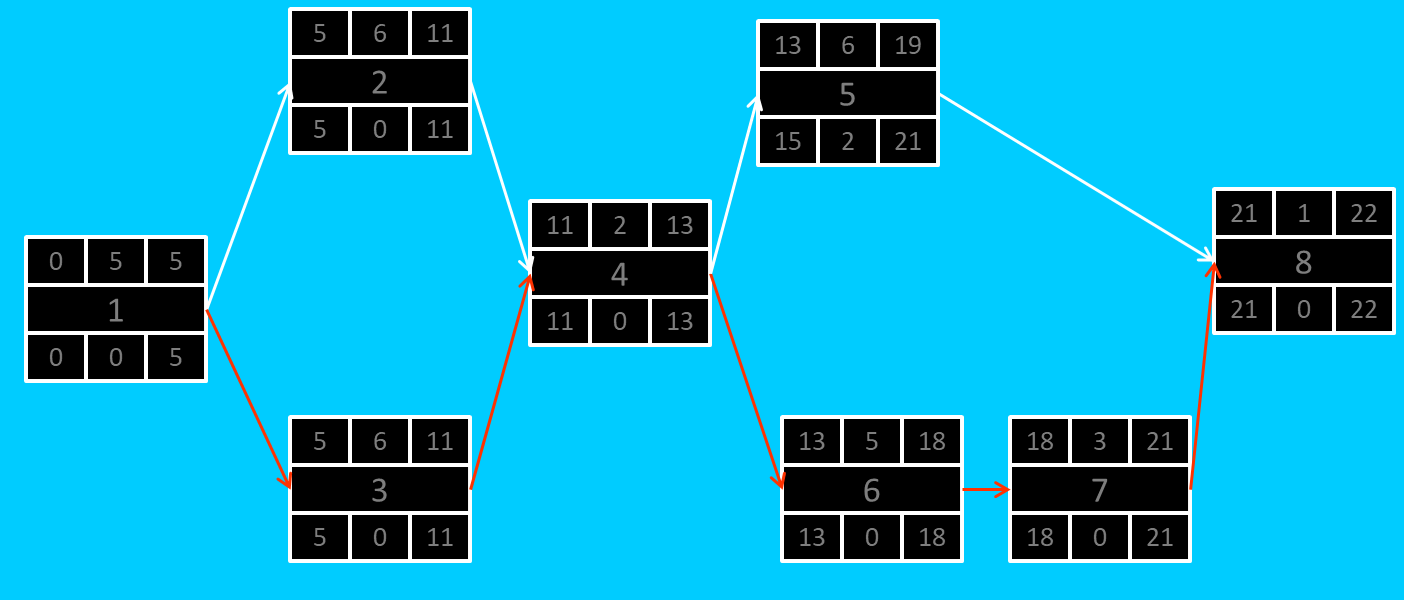
[Figure out the tasks required to complete your feature]

Example:

### Work items

| Task | Duration (PWks) | Predecessor Task(s) |
| --- | --- | --- |
| 1. Requirements Collection | 5 | - |
| 2. Screen Design | 6 | 1 |
| 3. Report Design | 6 | 1 |
| 4. Diagrams | 2 | 2, 3 |
| 5. Merge Screens | 6 | 4 |
| 6. Programming | 5 | 4 |
| 7. Testing | 3 | 6 |
| 8. Installation | 1 | 5, 7 |

### Pert diagram



### Gantt timeline

| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |