***RPG Project*** *<Name to be determined later>*

*Ian Beard*

[*ian.beard@bellevuecollege.edu*](mailto:ian.beard@bellevuecollege.edu)

*Michael Lablanc*

[*michael.lablanc@bellevuecollege.edu*](mailto:michael.lablanc@bellevuecollege.edu)

*John Sablaon*

[*John.sablaon@bellevuecollege.edu*](mailto:John.sablaon@bellevuecollege.edu)

***Scope***

I want to make a web based single player turn based RPG using HTML5 Canvas. The scope of the full project is potentially quite large. While perhaps not as massive as some notorious “90 hour RPGs,” this project could still be quite the undertaking

For the purposes of building a prototype, however, development will initially be limited to letting users play through a single battle.

Which would play out as selecting a group of characters. The computer generates a group of enemies. Then a character and enemies take turns, selecting actions to eventually defeat the other group until either the player defeats all the enemies and wins or loses when the enemy defeats all of the player’s.

A turn for a player’s character will start by them generating action points, which then will allow them to do one or more actions, such as attack or heal, until they are out of action points. At which point their turn is over, and the next character or enemy will get their turn.

***Business Narrative***

**General Specs**

- The game will be a single player turn based RPG. The prototype will mainly focus on the combat system.

- The game will use HTML5 Canvas for the visual elements. Pre-built game engines may already exist for HTML5, this will be researched early in development.

- The game will use Node.js and React (Subject to change)

- The database will use MySQL.

- The prototype will allow users to select both the player characters they want to use, and the enemies they would like to fight.

**Combat System**

Players will win after incapacitating all opponents. They will lose if all of their characters are incapacitated

Characters (playable or otherwise) have 6 main stats:

- Strength (determines physical attack damage)

- Willpower (determines strength of magical attacks and increases resistance to mental status ailments)

- Dexterity (determines accuracy of physical attacks)

- Focus (determines accuracy of magical attacks)

- Defense (reduces damage of incoming attacks and increases resistance to physical status ailments)

- Agility (reduces accuracy of incoming attacks)

Weapons will have a block rate and attack power assigned to them. Attacks will have damage values, accuracy ratings, and critical hit rates assigned to them.

Characters have 4 resources to manage during battle:

- Health (Reduced by taking damage from attacks, gained by healing. Characters are incapacitated when their health reaches 0.)

- Action Points (Characters gain a number of action points each turn, up to a certain limit. These points are spent to perform actions, like attacking or casting spells. Characters may take as many actions as they like per turn so long as they have enough points to spend <Limited to one attack per turn>.)

- Essence (Used to cast spells. Rather than a number of points being spent, Essence is accumulated by a character when they cast spells. If they continue to cast spells after they have reached their maximum amount of essence, they will take damage.)

- Tension (Represents a characters morale, measured as a percentage value. Tension is gained by scoring critical hits or from support spells, and is lost by missing, taking damage, or from certain enemy skills. High Tension increases accuracy and grants access to powerful actions, low tension lowers accuracy and prevents some actions from being performed. Tension also effects how resistant a character is to mental status ailments.)

Enemies will target characters based on their level of aggression (Aggro) towards them.

- Aggro towards each player controlled character is tracked separately by each enemy.

- Aggro towards a character is gained when they deal damage (only gained by the enemy taking damage), cast spells, or when they use abilities that explicitly state that it increase aggro.

- Aggro is lost when a character takes damage (only the aggro towards the character taking damage) or uses an ability that explicitly states that it decreases aggro.

- Players will be able to arrange their characters into a front and back battle line. Characters on the back line will gain aggro at a reduced rate, but will be unable to perform melee attacks.

***User Stories/Test Plan***

As a player, I need to be able to see which character and enemies I can select, so I can establish characters and enemies before starting the game.

Test: List of characters and list of enemies displayed on screen

As a player, I need to be able to select which characters and enemies I want to play with and against, so I can start the game.

Test: Characters/enemies can be added to list of selected characters/enemies

As a player, I need to be able to start the game after selecting the characters, so the game enters the gameplay.

Test: The screen displays the battle field and battle menu

As a player, I need to be able to view all characters and enemies on the screen, so I know that the game has started.

Test: The battle starts will all selected characters/enemies present on the screen

As a player, I need a visual indicator on the screen to reference character, or enemies turn so I know whose turn it is.

Test: Visual indicator for who’s turn it is should be on screen

As a player, I need to be able to pause and resume the game, so that the game will stop and continue as requested.

Test: Pause menu can be opened and closed at any time

As a player, and when the game is paused, I need to be able to restart the game at any point so that the current game will end and start a new game with the same characters starting with the initial state.

Test: The battlefield is reset to its initial state, with all selected characters and enemies on screen

As a player, I need to be able to exit the game at any point, so that I can select new characters and start the game over.

Test: The player is returned to the start screen

As a player, and when the game is paused, I need to be able to view the controls so I can make a selection

Test: The controls are displayed on screen

As a player, and when the game is paused, I need to be able to review the tutorial of the game mechanics.

Test: The Tutorial can be opened and closed via the pause menu while playing the game

As a player, I need to be able to access the tutorial from the selection screen, so that I can review the game mechanics.

Test: The tutorial can be opened and closed from the select screen

As a player, I need to be able to see the health of all characters and enemies on the screen, so I can strategize for my next play.

Test: A health bar can be seen below each character/enemy on screen

As a player, I need to be able to see which enemies are targeting which character, so I can anticipate the damage and prepare my next steps.

Test: A visual indicator for enemy aggro can be seen next to each enemy

As a player, I need to be able to view the characters action to perform, so that I can select one from the menu to execute.

Test: Available actions can be seen and selected via the battle menu

As a player, I need to be able to view the status of a character, so I can decide what my next move will be.

Test: Characters can be selected from the battle menu, which displays a popup showing their current status

As a player, I need to be able to select a valid target, so my character can perform the action selected to the target.

Test: Players should be able to see a visual indication of who they are targeting with their abilities

As a player, I need to be able to select when to end my turn based on available action points, so I can end my turn strategically.

Test: Performing a single action should not end the turn

As a player, I need to be able to view the status and resources of an active character, so I can decide what actions to perform in my next play.

Test: The “resources” of the active character should be seen next to the battle menu

As a player, I need to be notified with a victory screen when all enemies are defeated, so I know the game is over.

Test: A victory screen must be shown after winning a battle

As a player, I need to be notified when all of my characters are defeated with a defeat screen, so I know the game is over.

Test: A defeat screen must be shown after losing a battle

As a player, I need to be able to select to play the game again when it ends, so that the game will start over from the beginning.

Test: Players must have an option to return to the select screen after winning or losing.

***Use Case Scenarios***

|  |  |
| --- | --- |
| Player  Using Battle Menu | Check Character Stats |
| See Active Character Resources |
| See Active Character’s Available Actions |
| See Description/Cost of Available Actions |
| Select Action from List of Available Actions |
| Deselect Action Before Choosing Target |

***External Interfaces***

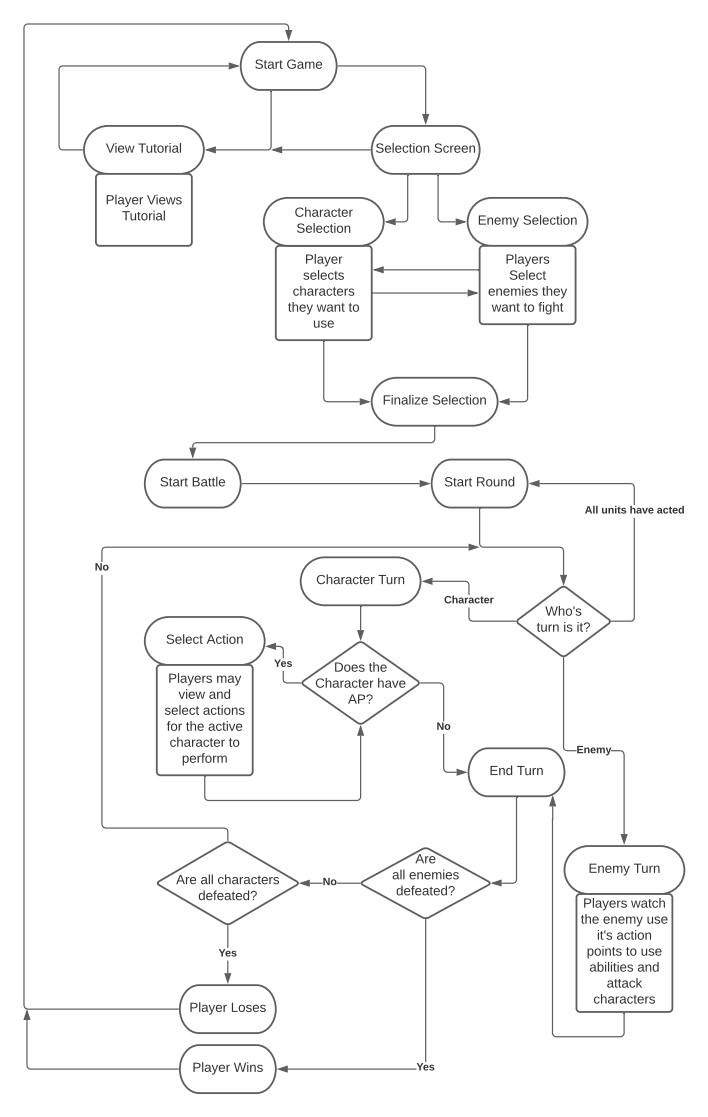
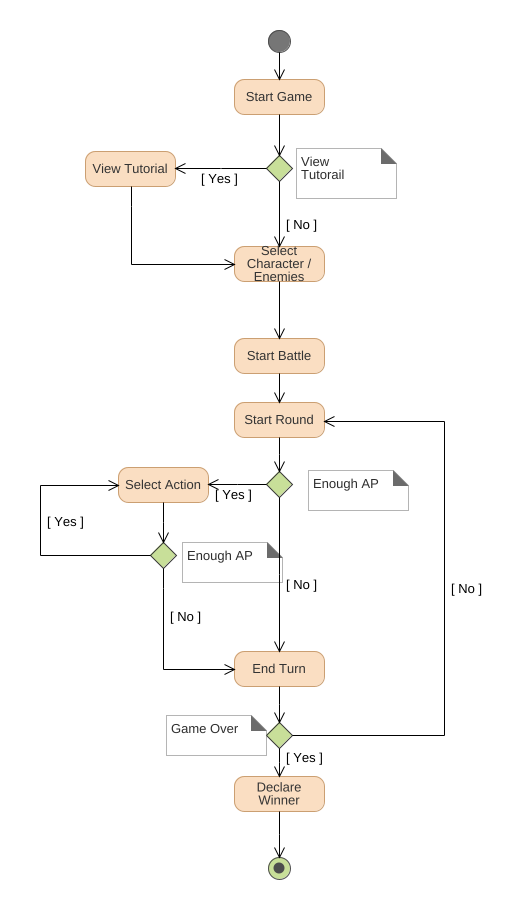
This system will use an external database to store information about characters and enemies. When the game is loaded, players will be provided with a list of characters and enemies to choose from that is populated by this database.

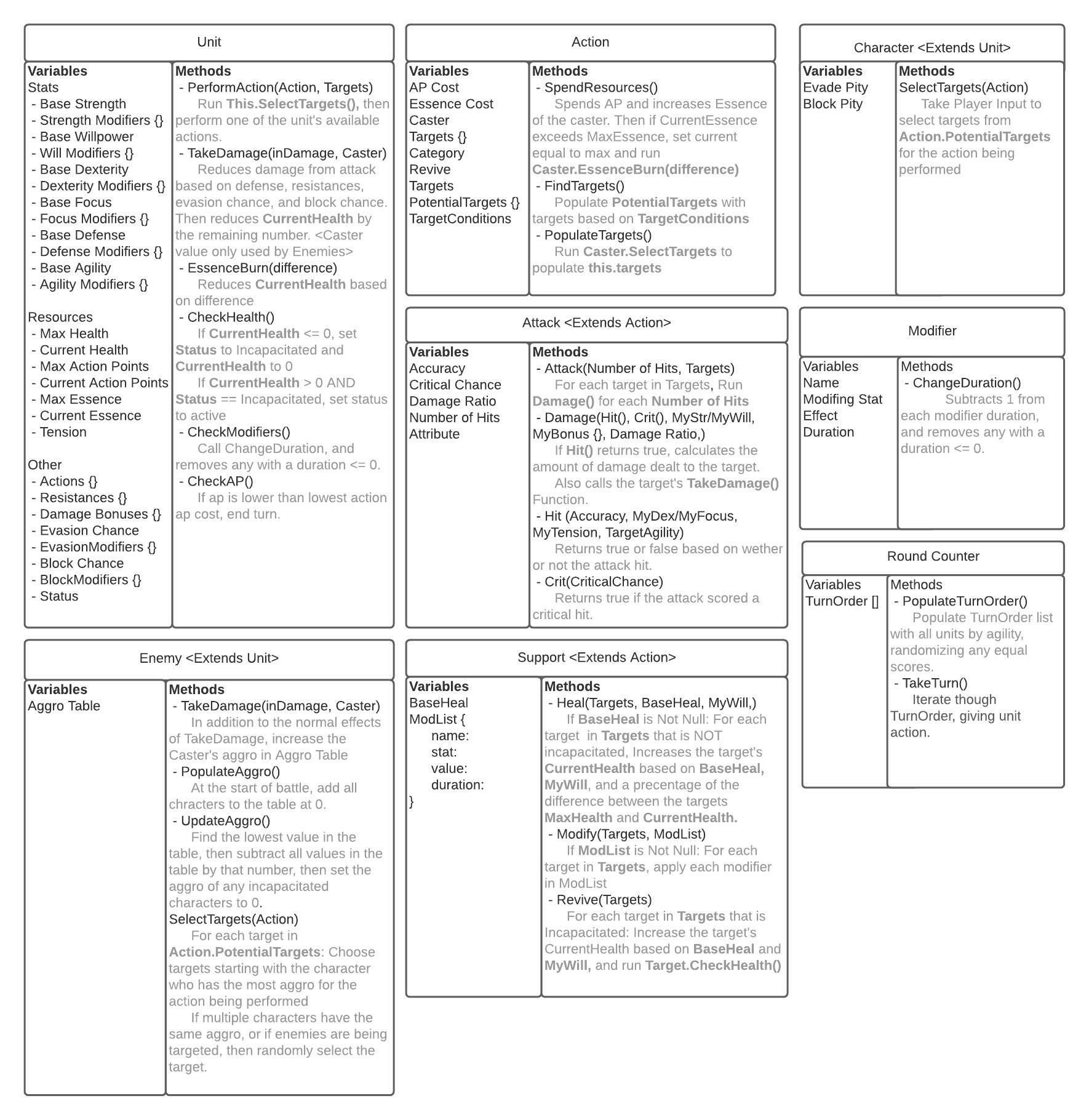
***API Calls***

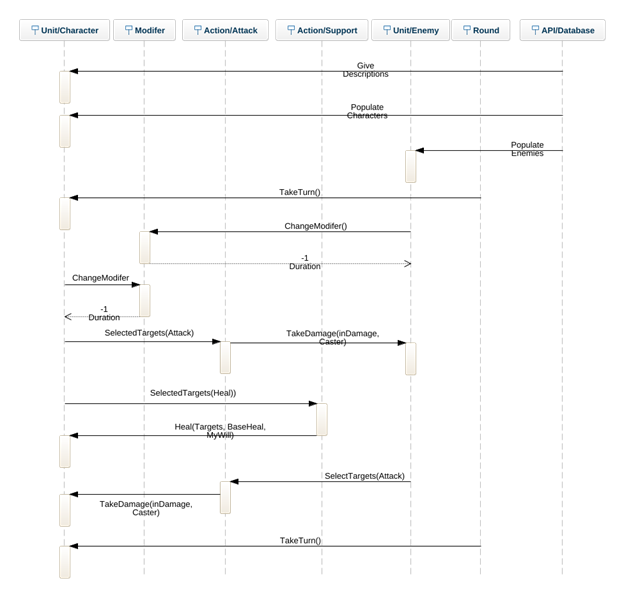
There will be one call when the program starts to populate the selection screen. It will retrieve names and descriptions from the database for the available characters and enemies.

Once the player has made their selections and starts the game, there will be a second call that retrieves all of the data for each character and enemy they selected.

Stretch Goal: There will be a login system for players so that they can store selections so that they can re-try previous battles. This will require sending and retrieving player data to and from the database.

***List of Steps/Activity Diagram***

***Class Diagram***

***Sequence Diagram***

***Class Responsibility Collaboration chart***

|  |  |
| --- | --- |
| **Unit Character** | |
| **Responsibility** | **Collaborators** |
| Start turn | Round |
| Check modifiers and lower duration | Modifier |
| Attack enemy | Action/ Attack |
| Heal party character | Action/ Support |
| Take damage to character | Action/ Attack |
| End turn | Round |

|  |  |
| --- | --- |
| **Modifier** | |
| **Responsibility** | **Collaborators** |
| Check modifiers and lower duration | Unit Character |
| Check modifiers and lower duration | Unit Enemy |

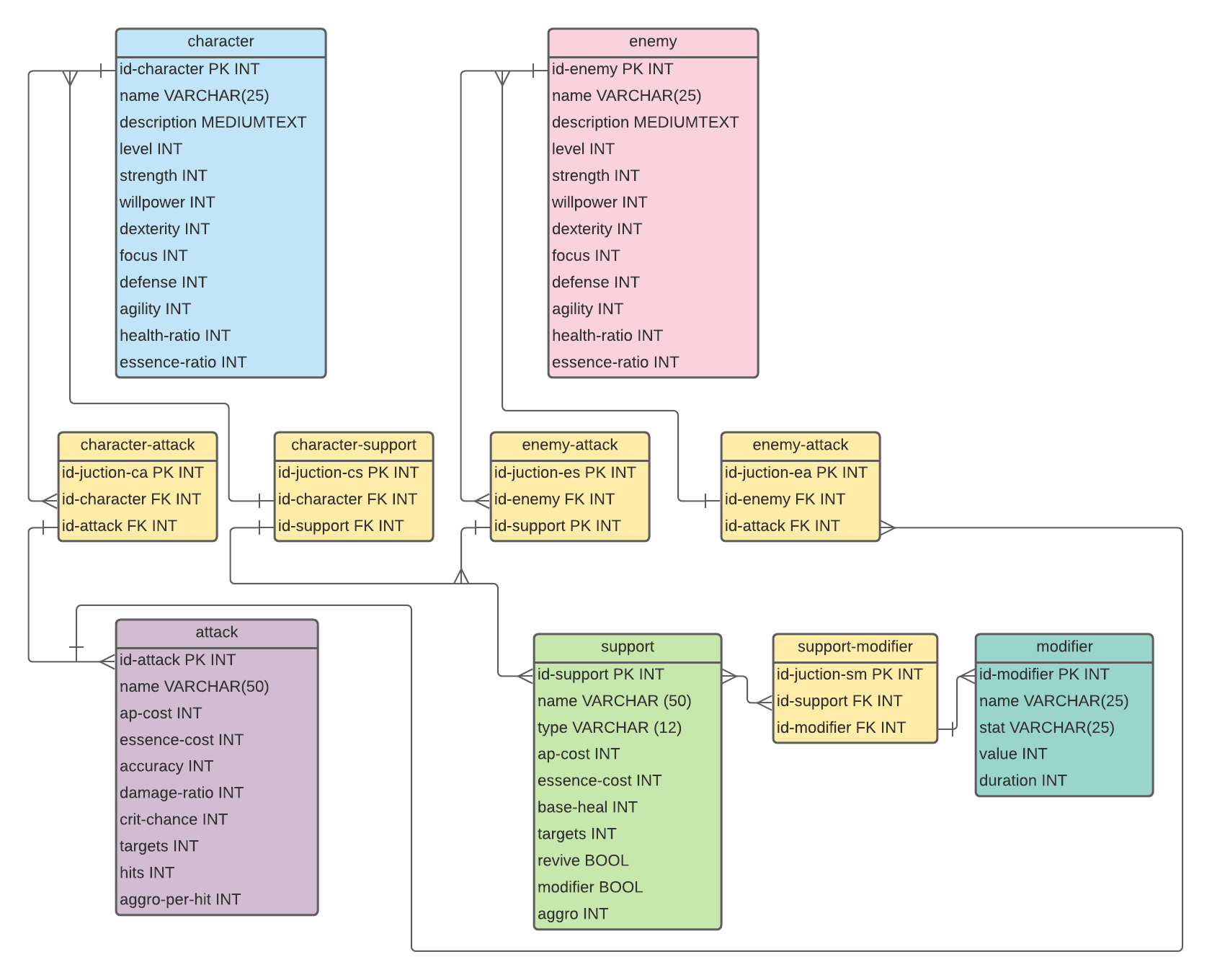
|  |  |
| --- | --- |
| **Action/ Attack** | |
| **Responsibility** | **Collaborators** |
| Attack Enemy | Unit Character |
| Attack Character | Unit Enemy |

|  |  |
| --- | --- |
| **Action/ Support** | |
| **Responsibility** | **Collaborators** |
| Heal party character | Unit Character |

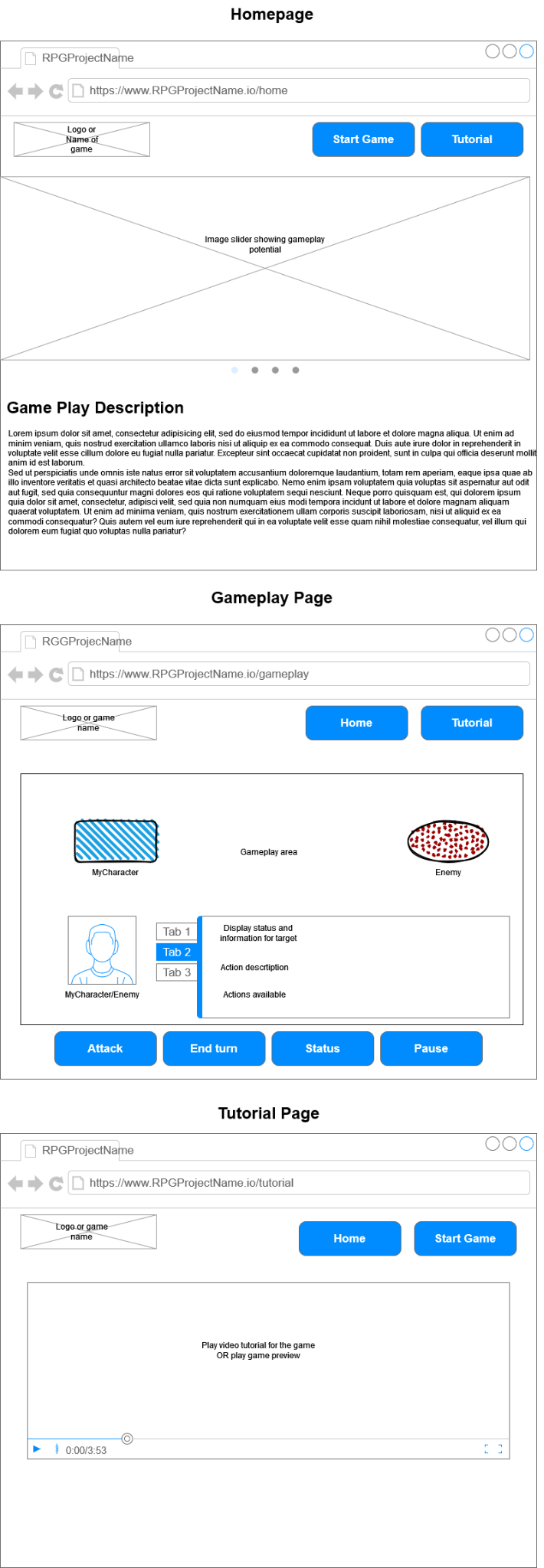
|  |  |
| --- | --- |
| **Unit Enemy** | |
| **Responsibility** | **Collaborators** |
| Start turn | Round |
| Check modifiers and lower duration | Modifier |
| Attack Character | Action/ Attack |

|  |  |
| --- | --- |
| **Round** | |
| **Responsibility** | **Collaborators** |
| Start turn | Unit Character |
| Start turn | Unit Enemy |

**Entity Relationship Diagram**

****

**Wireframe Diagram**

****

***Task Object Responsibility Chart***

|  |  |
| --- | --- |
| Object | Responsibility |
| Unit - Perform Action | Call functions from the selected **Action** class to perform said action |
| Unit - Essence Burn | Take a value passed from **Action.SpendResources()** and use it to calculate how much to reduce the unit’s **currentHealth** by. |
| Unit - Check Health | Any time the value of **currentHealth** changes, set **status** to the correct value:  - *good*: currentHealth/maxHealth > 0.5  - *low*: 0.5 >= currentHealth/maxHealth > 0.25  - *critical*: 0.25 >= currentHealth/maxHealth > 0  - *incapacitated*: currentHealth = 0 |
| Unit - Check Modifiers | Iterate through the **unit’s** list of **modifiers**, call that modifier’s **ChangeDuration** function, and then remove any **modifiers** that have a **duration** of 0 or less. |
| Unit - Check AP | Get the AP cost of each **action** in the unit’s list of actions, then force the unit to end it’s turn if it does not have enough AP to use any of those abilities. |
| Character - Select Targets | Using the input action’s **number of targets**, take user input to select targets for the action. |
| Character - Take Damage | Take an integer value passed from **Attack.Damage()** and apply values from the object calling **TakeDamage()** to modify the original amount. Then reduce the object’s **currentHealth** value. |
| Enemy - Populate Aggro | Add each **Character** participating in the battle and add them to the enemy’s **aggro table** |
| Enemy - Update Aggro | At the end of the **enemy’s** turn, find the **character** in the enemy’s **aggro table** that has the lowest amount of aggro. Then, subtract that amount from **all entries** in the table so that the **lowest value is set to 0** and thus becomes the new baseline value |
| Enemy - Select Targets | For each **target** from the input **action**, select the highest value in the enemy’s **aggro table** that has not already been selected as a target. Randomize tied values. |
| Enemy - Take Damage | Take an integer value passed from **Attack.Damage()** and apply values from the object calling **TakeDamage()** to modify the original amount. Then reduce the object’s **currentHealth** value. Then, use the attack’s **aggro per hit** value to update the caster’s entry in the aggro table. |
| Action - Spend Resources | Spend AP and Essence from the unit running the current action, and then run essence burn if the essence cost of the action exceeds their maximum essence |
| Action - Find Targets | Create a list of potential targets for the action |
| Action - Populate Targets | Use the unit’s select targets function to create a list of selected targets |
| Attack - Perform Attack | For each target in the list of selected targets, run the deal damage function a number of times equal to the attack’s specified number of hits. |
| Attack - Deal Damage | Run the attack’s hit and crit functions to determine the amount of damage dealt by the attack, and then pass that value on to the target’s take damage function. |
| Attack - Hit | Use the stats from the target and the unit performing the action to determine the hit chance for the atack, then run the helper class’s two random number function to determine whether or not the attack hit the target. Returns “miss” “hit” or “bonus” depending on the result. |
| Attack - Crit | Uses the helper class's single random number function to determine whether or not the attack scores a critical hit based on the attack’s critical chance. |
| Support - Heal | If the support action has a base heal value, then increase the target’s current health based on the action’s base heal, the caster’s willpower, and the target’s missing health (max health minus current health) |
| Support - Modify | If the support actions list of modifiers is not empty, add each modifier in the list to the target’s list of modifiers |
| Support - Revive | If the action allows targets to be revived, allow incapacitated targets to be healed using the support actions heal function. Otherwise, do not allow them to be targeted by the action. |
| Modifier - Change Duration | Have the modifier increase or decrease it’s own duration by the specified amount. |
| Game - Populate Turn Order | Sort every unit participating in the battle into a list based on their agility value. |
| Game - Take Turn | Iterate through the list generated by the populate turn order function. Have each unit in the list take their turn when they are called by this function. |
| Helper - One random number | Randomly generate an integer value between 1 and 100, and return the result *(return it in an array to be consistent with the output of two random numbers?)* |
| Helper - Two random numbers | Randomly generate two values between 1 and 100, and return an array containing both results. |