Mechanics Implementation

Moving Character:

* ‘WASD’ movement using Axes
* Apply movement using terrain navmesh and rigidbodies
* Play animation based on movement direction

Character Stamina:

* Stored as simple int or float that prevents rolling/attacking when too low
* Constantly regenerating and lowered by set amount when rolling
* Displayed in UI

Pickups:

* Prefabs that destroys self and runs a function when colliding with player
* Function determines the effect of each pickup

Breakables:

* Prefabs that destroys self and runs a function when colliding with player attack
* Function determines effect of being broken. May spawn pickups

Levers and switches:

* If activated by attack: Runs function when colliding with player attack
* If activated by key: Runs function when key is pressed with Player within a distance
* Function determines effect of each switch.
* Make the function it runs public. So each switch can be made to run a different function
* Play animation

Looting Chests:

* Runs function when key is pressed with Player within a distance
* Function determines effect of each Chest. May provide pickups or other effects
* Play animation
* Using either bool or state machine enum to prevent other actions while looting

Health:

* Stored as simple int or float that activates a death function when at or below 0
* Death function plays animation, respawns player and possibly resets level
* May include ‘Continue’ screen (basic Canvas interface)
* Health Displayed in UI

3rd person Camera:

* Use object parenting, mouse Axes and angle math to have camera rotate around Player based on mouse movement
* Use further angle math to prevent camera from looking too far up or down (inverting itself)
* Use further angle math to provide direction for player movement, attacks and lock-on

Camera Collisions with Environment:

* Use raycasting from player to determine the distance of the camera

Camera Lock-on:

* On key press find Array of enemies close to center of screen using either colliders or Camera.WorldToScreenPoint()
* If Array is not empty, find enemy closest to center and lock-on to that enemy
* Lock-on accomplished by setting a bool to true, which prevents mouse-based camera movement and uses Angle math to position the camera so that it is facing the enemy

Dodge Roll:

* Play animation
* Move Player over set distance (Applying acceleration over deltatime)
* Using either bool or state machine enum to prevent other actions while rolling

Attack:

* Play animation
* Using either bool or state machine enum to prevent other actions while attacking
* Test for sword – NPC collision and deal with result

Swapping ‘Weapons’:

* Have a canvas activate with buttons corresponding to each weapon
* Activate the buttons corresponding to owned swords
* Pressing buttons equips respective sword by replacing texture and setting “equipped”enum.
* Achieve “slow time” effect by using our own deltaTime variable that takes Time.deltatime and applies a modifier. Use this deltaTime for all player and npc calculations. Also slow animations.

Enemy AI Behaviours:

* Use State Machine enum to store enemy behaviours
* On spawn, set enemy to idle
* On trigger range from player, set enemy to pathing with player as target
* On attack range reached to player, set enemy to attack
* Boss AI needs to take into account boss moves and animations timers

Enemy Pathfinding:

* Uses navmesh and rigidbody to pathfind and interact with player or environment collision

Enemy Spawning:

* Enemies spawn in a large radius around player, so preset positions that spawn enemies for room when close enough etc.

Animations:

* Pair up animations provided by artists, and timing of animations with state machine of player.
* Make sure animations work as expected

Sound:

* Sound Effects easy to implement
* Narration also easy to implement
* Background Music Easy to implement