1. Executive Summary (1 page)

o Project title and team members with role assignments
o High-level overview of module purpose and game concept
o Key technical challenges anticipated.
Project title: Lunar Kingdom
Team Members: Haley Kloss, Hunter Beach, Mark Johnson
Role Assignments:

- Haley Kloss - Level Designer
- Hunter Beach - UI/UX Designer
- Mark Johnson - Tester

Key Tech Challenges:

1. In the gd extension, have three different processes with unique challenges
    a. the first- making sure that the bouncing physics are relative to fall velocity and object size
        - Calculating the velocity
        - Calculating the mass
        - Ground texture
    b. The second - make star fall object type that sends out kill signal when touched by player
    c. The third -emits custom signals on bounce and fade-out events to trigger particles, sound, and cleanup
2. Level design - Ensuring that there is enough content to fill at least 3-5 minutes of gameplay
3. Ensuring that all code is consistent in its naming conventions, format, and documentation with three people working on the code base

The following document outlines the Group 4 Final Project, *Lunar Kingdom*, by Haley Kloss, Hunter Beach, and Mark Johnson. Haley Kloss will be acting as the level designer,

2. Module Specification (3-4 pages)
o Detailed description of module functionality
o Technical architecture and design patterns (singleton, observer, factory etc.)
o API documentation (classes, methods, parameters)
o Integration approach with the Godot engine
o Comparison with existing solutions (if applicable)

Module Functionality:

- Overrides the collision response of a 2D body and a ‘ground’ so that the body bounces
- **Dynamic Bounce Strength**: The strength of the bounce will be calculated dynamically
  - Take into account impact velocity, object size, and ground material
  - Formula: bounce_strength = base_strength * (velocity * size * surface_factor)
- **Energy Loss**: Make the bounce slow down and stop realistically over time
  - Each bounce, a percentage of energy is lost until there is none left and bouncing stops
  - Make an exposed variable for energy_loss_rate so that it can be adjusted in editor
- **Surface Material**: One factor will be ground material, which can be registered for surfaces, with the softer the ground the less the bounce (slider)
- **Error handling:** checks for no mass, no velocity, no ground texture, handles these and logs warning in console
- **Star Object:** make star fall object type that sends out kill signal when touched by player
- **Impact Events**: emits custom signals on bounce and fade-out events to trigger particles, sound, and dissipation when energy runs out.
- **Purpose**: Produces more realistic physics for falling objects in platform style games by allowing objects land more realistically, with a bounce, instead of falling and becoming immediately static. This adds visual interest, as well as an additional challenge if applied to obstacle or enemy, forcing the player to account for a bounce when timing their attack or evasion.

Technical Architecture and Design Patterns:
Classes
BounceNode2D: inherits from RigidBody2D or Node2D
SurfaceMaterial: a resource that holds the coefficient for a surface’s hardness
BounceResources: used to register factors for bounces
StarNode2D: an object that has gravity, has bounce applied to it, and kills the player on impact

Design Patterns:
Observer: The impact or dissipation of an object triggers particle and sound effects, and as a result invokes the observer pattern. The star object also emits a kill signal on impact with the player.
Factory: The factory method is used in this module to generate SurfaceMaterial objects, as they will not all be the same.

Singleton: The singleton design pattern is present in shared constants such as gravity and default values for the bounce equation.

API Documentation:

Falling Hazard Process Overview:

1. Detect a collision between the ground and a DynamicBounceBody2D
2. Override the _integrate_forces, calculate dynamic bounce, and apply bounce to the object
3. The signal class triggers impact events such as particles on impact
4. Repeat the process until the energy has dissipated

DynamicBounceBody2D: a custom node that extends RigidBody2D, creating an object that bounces dynamically when it collides with the 'ground.' This node has custom properties that are visible in the editor, including size, energy, energy loss, base strength, and fade on stop.

Surface Material: A new resource that stores and applies surface material properties for hardness to allow for custom bounce behavior on it. This resource is built in as a property of

Impact Events: A class that emits custom signals when a collision occurs or when the object dissipates after losing all of its bounce energy. It will trigger particles and sounds. If bool fade_on_stop is true, the object in question will be removed from the game once the dissipation particles and sound are completed.

HazardObject: A custom node that extends a RigidBody2D, this object has gravity applied, is integrated with bounce and impact events, and sends out a kill signal when the player touches it.

Properties Accessible in Editor:

Float size

Float energy

Float energy_loss

Float base_strength Bool fade_on_stop

Property surface_material: Float hardness

Property impact_event

Integration approach with Godot Engine:
Implemented as a GDExtension, not a built-in module

Compiled as a shared library (.gdextension + .dll/.so)

Classes registered with GDREGISTER_CLASS()

Methods and signals exposed through ClassDB::bind_method() and ADD_SIGNAL()

Integrates seamlessly into existing Godot physics — overrides collision response without replacing engine core

3. Game Design Document (3-4 pages)
o Game concept and genre
o Core gameplay mechanics
o How the custom module enhances gameplay
o Target audience and platform
o Visual style and aesthetic direction

Game Concept and Genre:
The game will be a 2D platformer with a pixelated fantasy art style. In it, you are a satyr (https://lucky-loops.itch.io/character-satyr) making his way through the woods of the Lunar Kingdom to get to the castle at the end. The way will have difficult jumps, enemies, and items to collect in order to succeed, such as the key to the castle. There will also be other obstacles, such as deadly stars that rain down from the sky and bounce on the ground until they dissipate, which will kill the player on contact. To win, the player must collect the key and make it to the castle doors without falling off the map, being killed by an enemy, or being killed by a falling star.

Core Gameplay Mechanics:
Player: the player will be able to idle, run, jump, crouch, and pick up items. Idle will occur when the user is not causing movement or falling, run will be linked to both the left and right arrow keys as well as the A and D keys. Jump will be linked both to W and the space bar. Crouch will be linked to the down arrow and the S key.

The player must jump from platform to platform to make their way to items and the end of the level; falling off will cause game over.

There will be an enemy patrolling on one of the platforms, and will lock in on and follow the player when in range. Touching the enemy will kill the player, causing a game over.

Falling stars and enemies must be avoided, as touching either will kill the player, causing a game over.

To win the game, the player must collect the key by touching it, then bring the key to the castle door at the end of the level. Touching the door with the key in the player's inventory causes a game win.

How the custom module enhances gameplay:

The custom module will enhance the game in terms of gameplay by causing more realistic movement of obstacles. This extra movement makes it more difficult for the player to evade these obstacles than if they were static once reaching the ground. This additional challenge adds a level of interest and skill that the game would not otherwise exhibit.
Not only that, but the module also increases the visual appeal of the game. Having moving features instead of just a fixed landscape makes the game more interesting to look at.

Target audience and platform:
The target audience for this game is people 8 years and older. There is no graphic content or adult themes, and as such is only age-restricted due to difficulty level. The simplicity of a platformer also appeals to a broad range of age groups and gaming skills, as they are extremely common and easy to grasp.

The intended platform is PC, as the game is wired for keyboard input.

Visual style and aesthetic direction:
The visual style of the game will be 2D pixel art. The style will incorporate nature and fantasy elements, and be set at night. The feeling evoked by the game should be cozy, nostalgic, and a tiny bit creepy, with cute but odd creature sprites and a player character.

4. Technical Implementation Plan (2-3 pages)
o Development timeline with milestones
o Technology stack and dependencies
o Testing strategy and success metrics
o Risk assessment and mitigation strategies

Development timeline with milestones:
Deliverable 1: Oct 28, 2025 12:00am until Nov 18, 2025 11:59pm
Deliverable 2: Oct 28, 2025 12:00am until Dec 4, 2025 11:59pm

1. Nov. 16: Base game - background, platforms, tilemap
2. Nov. 19: GUI - start menu, game over, game start, key in inventory bool, need key popup
3. Nov. 23: Enemy and Player movement
4. Nov. 25 Falling stars- fall, can kill the player
5. Nov. 26: Module Completed - apply to falling stars so that they bounce, then disappear when energy is depleted
6. Nov. 3: Finish testing, submit

Technology stack and dependencies:
Godot, C#
Testing strategy and success metrics:
Strategy: Play through the game and ensure all benchmarks are met:

1. Touching an enemy causes game over
2. Touching a fallen star causes game over
3. Falling off the map causes game over
4. Getting to the end without the key causes a GUI pop-up, not a game win
5. Getting to the end with the key causes the game to be won
6. Play time is at least 3 minutes
7. Player responds accurately to keyboard input
8. All platforms can be walked on without issues
9. No jumps that are meant to be performed are impossible or too difficult to complete in 5 tries or less
10. When the player dies, a game over screen pops up, then the main menu pops up after 2 seconds
11. When the player wins, a game won screen pops up, then the main menu pops up after 2 seconds
12. Each time the player dies or wins, the game is restarted, with no falling stars on the ground and the key out of the player inventory and back in the map

Risk assessment and mitigation strategies:
Low risk
Have falling stars default to dissipating after 5 sec

5. Team Collaboration Plan (1-2 pages)
o Role distribution and responsibilities
o Communication protocols and meeting schedule
o GitHub workflow (branching strategy, code review process)

Role distribution and responsibilities:
Haley Kloss -
Hunter Beach -
Mark Johnson -

Communication protocols and meeting schedule:
Discord is the primary communication forum
Meet through voice chat every ___ at ___

GitHub workflow (branching strategy, code review process):
Every member will have their own branch, which should be reviewed by another team member and merged into the main branch after each major component being worked on is complete and tested.

References and Resources (1 page)
o Academic and technical sources
o Tutorials and documentation consulted
o Inspiration and prior art

Academic and technical sources:

Tutorials and documentation consulted:

1. https://docs.godotengine.org/en/4.4/getting_started/first_2d_game/index.html
2. https://www.youtube.com/watch?v=93QBVvCzUGI&list=PLhXFaKLHQJdXpwaNt6gGwpHLTWL0m-TSL&index=9

Inspiration and prior art:
Player Character: https://lucky-loops.itch.io/character-satyr