Black text: original proposal content
Blue text: how Milestone 1 aligns with the proposal

Technical Elements:

Identify how the game satisfies the core technical requirements: rendering; geometric/sprite/other assets; 2D geometry manipulation (transformation, collisions, etc.); gameplay logic/AI, physics.

- **Rendering**: Render pixel art onto predetermined maps. Render changes to the map (e.g. terrain breaking apart) when these changes are triggered by player actions.
    - Milestone 1:
        - Currently, textured used are temporary normal 2D graphics instead of pixel arts for testing. The art style will be changed into pixel art in the future.
- **Geometric/sprite/other assets**: Playable characters, enemy characters, weapons (projectiles or melee) are sprites. Character animation (movement, actions), clickable buttons in the menu overlay (abilities, character 1/2/3...) are assets.
    - Milestone 1:
        - Playable characters and enemy characters are sprites.
        - The attack effect is temporarily generated with a red square.
        - Camera can be moved using arrow keys within the map boundary while the overlay UI buttons will stay fixed on the screen,
- **2D Geometry manipulation:** Player and enemy sprites will be translated to move around the map for each level. Collisions between sprites will be detected. Character and weapon sprites will be animated to show movement and attacks.
    - Milestone 1:
        - Playable characters can move around the map using A and D keys to move left and right respectively.
        - Basic collisions between characters and terrains could be detected. If a player or enemy collides with the terrain, it will not be able to move to the direction of the terrain.
        - When a character is attacked, its sprite will flash red and shake.
- **Gameplay logic:** Need logic to determine outcome of character movements and attacks, allow characters to interact with weapons and potion ingredients, and to determine when the player team has succeeded or failed to complete the level. The order of the attacks will be determined based on the character's stats.
    - Milestone 1:
        - The collision between the attack and sprites will be used to determine if an attack is successful and damage will be taken.

        - The order of the attacks is determined based on the character's initiative value.
- **Physics:** Player and enemy characters can launch projectiles at one another, which will be affected by gravity and launch angle. Different terrains (e.g. chocolate, gummy) will affect characters' movement abilities on a particular map.
    - Milestone 1:
        - Mouse click could determine the attack angle.
- **AI**: Need a search algorithm (BFS) and/or a basic decision tree for the enemy behavior. Enemies move to the closest player target in order to get into actionable range. Depending on stats like location and health, the enemy chooses appropriate action.
    - Milestone 1:
        - The enemy has random movement but always attack to the left.
- **Audio:** Sound effects for player and enemy actions (e.g. attacks) or other gameplay events (e.g. beating a level).

Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

- Story elements
    - Animation for the storytelling with text dialogue
- Parallax scrolling background
    - Alternative: static scrolling background
    - Impact would be purely aesthetic if excluded
- Enemy group behavior cooperative planning
    - When multiple enemies are within a certain range, they will work together to "gang up" on a single player character
    - Alternative: enemies will act independently regardless of their proximity to one another
    - Gameplay would be relatively unaffected if this element is excluded; enemy teams may become easier to defeat since it would be possible for individuals to act against the team's best interests
- Complex physical interactions with the environment
    - Breaking/changing terrain during gameplay
    - Alternative: no terrain changes during gameplay
    - Gameplay would be less complex if this element is excluded, but the game would still be playable

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

- Keyboard and mouse input
    - Milestone 1:
        - Mouse input for attack angle
- W (climb up), A (move left), S (climb down), D (move right)
    - Milestone 1:
        - A (move left), D (move right)
- Mouse left click to interact with menu overlay (like clicking on skills or characters)
    - Milestone 1:
        - Mouse input for button
- Use arrow keys to move the camera to see the other parts of the level
    - Milestone 1:
        - Arrow keys can move the camera
- Angle arrow when using a skill that fires a projectile will follow mouse placement. Left click to fire

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Week: January 24 - **Milestone 1 Sprint Start**

- First level design
- ECS model (Add a player character and an enemy character)
- Basic 2D rendering and rendering effects
- Basic character movement (WASD)

Week: January 31 - **Skeletal Game (due on February 4, Friday) (for creative part, 2 basic or 1 advanced)**

- Basic physics (e.g. gravity)
- Enemies' movement is initially random
- Overlay for abilities (melee attacks only)
- Simple texture for one playable character and basic terrain texture (textured geometry)
- Key-frame/state interpolation of playable character movement
- Well-defined game space boundaries for the designed first level
- Correct collision processing

- Camera controls

Milestone 1:

- Implemented:
    - A basic ECS model is implemented with components such as player, enemy, terrain, health, motion, camera, background, etc.
    - Have basic 2D rendering with simple textures
    - Have basic character movement with A (left) and D (right)
    - Key-frame and state are interpolated by setting different states for characters
    - Game space boundaries are created using terrains
    - Basic collision detections are implemented
    - Camera controls using arrow keys are implemented
    - Simple randomized enemy movement and a hard-coded attack action
- Missing:
    - A (up) and S (down) controls are missing due to the map design; the map of the first level does not require these movements

Week: February 7 - **Milestone 2 Sprint Start**

- Sprite sheet animation
- Projectile abilities angle preview
- Simple decision tree for enemy abilities
    - One ranged attack, one melee attack
    - End turn if no player characters are in range for attacks
- Help/tutorial features for users

Week: February 14

- New integrated assets
- Basic user tutorial/help
- More level design
- Advanced physics and animation
    - Projectile attacks

Week: February 21 - **Minimal Playable Game (due on February 25, Friday)**

- Reloadability and level selection
- Debugging earlier milestones

Week: February 28 - **Milestone 3 Sprint Start**

- Debugging earlier milestones
- Additional levels/terrains
- Handle user input

- Parallax scrolling background

Week: March 7

- Memory management
- Real-time gameplay
- Audio effects

Week: March 14 - **Playable Game (due on March 18, Friday)**

- Debugging earlier milestones
- Consistent game resolution
- Story refinement
    - Introductory animations
    - Ending animations

Week: March 21 - **Milestone 4 Sprint Start**

- Debugging earlier milestones
- Physics-based animations
- Comprehensive tutorial levels
- Enemy group behavior

Week: March 28

- Main menu screens (main menu, level select, character select, character stats screen)
- Optimize user interaction
- Complete tutorial and additional levels
- Evaluate and improve UX

Week: April 4 - **Final Game (due on April 8, Friday)**

- Complete additional advanced features as necessary
- Debugging and minor improvements as necessary