<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">

<title>Geminus - Dark Fantasy RPG (GDD Integrated)</title>

<script src="https://cdn.tailwindcss.com"></script>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600&family=Orbitron:wght@400;700;900&display=swap" rel="stylesheet">

<style>

/\* --- Molten Core Theme --- \*/

:root {

--highlight-orange: #f97316;

--glow-red: #ef4444;

--text-primary: #f0f0f0;

--text-secondary: #a0a0a0;

--panel-bg: rgba(20, 20, 22, 0.75);

--input-bg: rgba(0, 0, 0, 0.4);

--border-color-main: rgba(249, 115, 22, 0.4);

--border-color-pulse: rgba(239, 68, 68, 0.7);

/\* --- Combat & UI Colors (PRESERVED) --- \*/

--color-text-health: #00BFFF; /\* 🔵 \*/

--color-text-exp: #0BDA51; /\* 🟢 \*/

--color-text-next-level: #BF00FF; /\* 🟣 \*/

--color-text-drop: #FF7518; /\* 🟠 \*/

--color-text-level: #00BFFF; /\* 🔵 \*/

--color-text-gold: #EFBF04; /\* 🟡 \*/

--color-text-location: #FF1D15; /\* 🔴 \*/

--color-status-ok: #7CB342;

--color-status-warn: #FDD835;

--color-status-danger: #D32F2F;

--hp-color: #E53935;

}

@keyframes pulse-border {

0%, 100% {

border-color: var(--border-color-main);

box-shadow: 0 0 15px rgba(249, 115, 22, 0.2), 0 4px 30px rgba(0, 0, 0, 0.6);

}

50% {

border-color: var(--border-color-pulse);

box-shadow: 0 0 25px rgba(239, 68, 68, 0.4), 0 4px 30px rgba(0, 0, 0, 0.6);

}

}

@keyframes pulse-main-tab-glow {

0%, 100% {

border-bottom-color: var(--highlight-orange);

text-shadow: 0 0 8px var(--highlight-orange);

}

50% {

border-bottom-color: var(--glow-red);

text-shadow: 0 0 12px var(--glow-red);

}

}

html, body { height: 100%; overflow: hidden; }

body { font-family: 'Inter', sans-serif; color: var(--text-primary); background-color: #121212; -webkit-user-select: none; user-select: none; touch-action: none; }

.font-orbitron { font-family: 'Orbitron', sans-serif; color: var(--text-primary); text-shadow: 0 0 8px rgba(249, 115, 22, 0.5); }

.glass-panel {

background: var(--panel-bg);

-webkit-backdrop-filter: blur(10px);

backdrop-filter: blur(10px);

border: 1px solid;

animation: pulse-border 4s ease-in-out infinite;

}

.glass-button {

background: rgba(255, 255, 255, 0.05);

border: 1px solid rgba(249, 115, 22, 0.25);

color: var(--text-primary);

transition: all 0.2s ease-in-out;

display: flex; align-items: center; justify-content: center; flex-shrink: 0;

font-family: 'Orbitron', sans-serif;

text-shadow: 0 0 5px rgba(249, 115, 22, 0.3);

}

.glass-button:not(:disabled):hover {

background: rgba(249, 115, 22, 0.15);

border-color: rgba(249, 115, 22, 0.6);

transform: translateY(-2px);

color: var(--highlight-orange);

text-shadow: 0 0 10px var(--highlight-orange);

}

.glass-button:disabled {

background: rgba(60, 60, 60, 0.4) !important; border-color: rgba(100, 100, 100, 0.8) !important;

cursor: not-allowed; color: #6b7280 !important; text-shadow: none !important; box-shadow: none !important; animation: none;

}

.glass-button:not(:disabled):active { transform: translateY(1px) scale(0.98); }

.progress-bar-track { background-color: rgba(0,0,0,0.5); border-radius: 9999px; overflow: hidden; border: 1px solid rgba(0,0,0,0.7); }

.progress-bar-fill { border-radius: 9999px; height: 100%; transition: width 0.3s ease-out; }

.custom-scrollbar::-webkit-scrollbar { width: 6px; }

.custom-scrollbar::-webkit-scrollbar-track { background: transparent; }

.custom-scrollbar::-webkit-scrollbar-thumb { background-color: rgba(249, 115, 22, 0.4); border-radius: 3px; }

.custom-scrollbar-x::-webkit-scrollbar { height: 4px; }

.custom-scrollbar-x::-webkit-scrollbar-track { background: transparent; }

.custom-scrollbar-x::-webkit-scrollbar-thumb { background-color: rgba(249, 115, 22, 0.4); border-radius: 2px; }

#main-content { transition: transform 0.4s ease-in-out, opacity 0.4s ease-in-out; }

#main-content.focused { position: fixed; inset: 0; z-index: 50; width: 100vw; height: 100vh; border-radius: 0; padding: 0; }

.main-tab-button {

flex-shrink: 0;

font-family: 'Orbitron', sans-serif;

font-size: 0.8rem;

padding: 0.75rem 1rem;

background: transparent;

border: none;

border-bottom: 3px solid transparent;

color: var(--text-secondary);

cursor: pointer;

transition: color 0.3s;

}

.main-tab-button.active {

color: var(--highlight-orange);

animation: pulse-main-tab-glow 4s ease-in-out infinite;

}

.main-tab-panel { display: none; }

.main-tab-panel.active { display: block; }

.modal-backdrop { position: fixed; inset: 0; background-color: rgba(0, 0, 0, 0.8); display: flex; align-items: center; justify-content: center; z-index: 60; padding: 1rem; }

#toast-notification { position: fixed; bottom: -100px; left: 50%; transform: translateX(-50%); padding: 12px 24px; border-radius: 8px; font-weight: 600; transition: bottom 0.5s ease-in-out; z-index: 100; border: 1px solid; }

.toast-error { background-color: rgba(255, 80, 80, 0.2); border-color: #ff5050; color: #ff5050; }

.toast-success { background-color: rgba(249, 115, 22, 0.2); border-color: var(--highlight-orange); color: var(--highlight-orange); }

#smoke-canvas { position: fixed; top: 0; left: 0; width: 100%; height: 100%; z-index: -1; pointer-events: none; opacity: 0.5; }

.equipment-grid { display: grid; grid-template-columns: 1fr 1fr; gap: 0.5rem; }

.equipment-slot-wrapper { background-color: transparent; border-radius: 0.5rem; padding: 0.5rem; border: 2px solid transparent; }

.equipment-slot-title { display: flex; align-items: center; gap: 0.5rem; font-weight: 600; color: var(--text-primary); margin-bottom: 0.5rem; font-size: 0.8rem; }

.equipment-slot-content { display: flex; align-items: center; justify-content: center; gap: 0.5rem; border-radius: 0.375rem; min-height: 60px; padding: 4px; position: relative; background-color: transparent; cursor:pointer; border: 1px solid transparent; transition: all 0.2s; }

.equipment-slot-content:hover { border-color: var(--highlight-orange); }

.inventory-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(60px, 1fr)); gap: 0.5rem; }

.inventory-slot { position: relative; width: 100%; padding-bottom: 100%; height: 0; /\* Aspect ratio fix \*/ border: 2px solid transparent; border-radius: 0.375rem; background: transparent; cursor: pointer; transition: all 0.2s; }

.inventory-slot:hover { border-color: var(--highlight-orange); }

.inventory-slot.selected { border-color: var(--glow-red); box-shadow: 0 0 10px var(--glow-red); }

.inventory-slot img { position: absolute; top: 0; left: 0; width: 100%; height: 100%; object-fit: contain; padding: 4px; }

.item-label { position: absolute; bottom: 2px; right: 2px; background: rgba(0,0,0,0.7); font-size: 0.6rem; padding: 1px 3px; border-radius: 3px; }

.gem-pouch-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(52px, 1fr)); gap: 0.5rem; }

.gem-item { position: relative; border: 2px solid transparent; border-radius: 0.375rem; background: transparent; cursor: pointer; aspect-ratio: 1 / 1; display: flex; align-items: center; justify-content: center; transition: all 0.2s; }

.gem-item:hover { border-color: var(--highlight-orange); }

.gem-item.selected { border-color: var(--glow-red); box-shadow: 0 0 10px var(--glow-red); }

.stat-accordion-item { background: rgba(0,0,0,0.2); border-radius: 0.5rem; margin-bottom: 0.75rem; overflow: hidden; border: 1px solid rgba(249, 115, 22, 0.15); }

.stat-accordion-header { display: flex; justify-content: space-between; align-items: center; width: 100%; padding: 0.75rem 1rem; background: rgba(249, 115, 22, 0.05); cursor: pointer; }

.stat-accordion-header h3 { font-family: 'Orbitron', serif; font-size: 1.1rem; color: var(--text-primary); }

.accordion-arrow { transition: transform 0.3s ease; }

.stat-accordion-item.open .accordion-arrow { transform: rotate(90deg); }

.stat-accordion-content { max-height: 0; overflow: hidden; transition: max-height 0.4s ease-out, padding 0.4s ease-out; padding: 0 1rem; }

.stat-accordion-item.open .stat-accordion-content { max-height: 1000px; padding: 1rem; }

.stat-line { display: flex; align-items: center; padding: 0.5rem 0; font-size: 0.9rem; border-bottom: 1px solid rgba(249, 115, 22, 0.1); }

.stat-line:last-child { border-bottom: none; }

.stat-icon { font-size: 1.1rem; margin-right: 0.75rem; width: 20px; text-align: center; }

.stat-name { flex-grow: 1; }

.stat-value { font-weight: 600; color: #fff; }

.info-btn { background: none; border: 1px solid var(--text-secondary); color: var(--text-secondary); border-radius: 50%; width: 20px; height: 20px; font-size: 0.7rem; line-height: 18px; text-align: center; cursor: pointer; margin-left: 0.75rem; }

.attr-btn { background: rgba(249, 115, 22, 0.2); border: 1px solid var(--border-color-main); color: var(--highlight-orange); border-radius: 6px; width: 28px; height: 28px; font-weight: bold; cursor: pointer; transition: all 0.2s; margin-left: 0.5rem; font-size: 1.2rem; }

.attr-btn:hover:not(:disabled) { background: rgba(249, 115, 22, 0.4); }

.attr-btn:disabled { background: rgba(75, 85, 99, 0.2); border-color: #4b5563; color: #6b7280; cursor: not-allowed; }

@keyframes flash { 0% { color: #fff; text-shadow: 0 0 10px #fff; } 50% { color: var(--highlight-orange); text-shadow: 0 0 15px var(--highlight-orange); } 100% { color: #fff; text-shadow: none; } }

.flash-update { animation: flash 0.5s ease-out; }

#stat-info-modal { position: fixed; inset: 0; z-index: 1001; background-color: rgba(0,0,0,0.7); display: none; align-items: center; justify-content: center; }

#stat-info-backdrop { position: absolute; inset: 0; }

.stat-info-content { background: var(--panel-bg); border: 1px solid var(--border-color-main); padding: 1.5rem; border-radius: 0.5rem; max-width: 300px; text-align: center; }

.game-key {

display: flex; align-items: center; justify-content: center;

font-family: 'Orbitron', sans-serif; font-weight: 900;

color: var(--highlight-orange); background: var(--panel-bg);

border: 2px solid var(--border-color-main); border-radius: 8px;

cursor: pointer; user-select: none; transition: all 0.1s ease-in-out;

text-shadow: 0 0 8px var(--highlight-orange);

}

.move-key { width: 40px; height: 40px; font-size: 20px; }

.move-key svg { width: 1.2em; height: 1.2em; fill: currentColor; }

#d-pad-controls { display: grid; grid-template-areas: ". up ." "left down right"; gap: 5px; }

#key-up { grid-area: up; } #key-left { grid-area: left; } #key-down { grid-area: down; } #key-right { grid-area: right; }

#key-interact { width: 130px; height: 40px; font-size: 16px; font-weight: 700; }

.game-key.pressed { background: rgba(10, 10, 12, 0.85); border-color: var(--glow-red); transform: scale(0.95); box-shadow: 0 0 15px var(--glow-red), 0 0 20px rgba(0,0,0,0.6) inset; color: var(--glow-red); }

#player-status-container .level-label { color: var(--text-secondary); }

#player-status-container .name-value { color: var(--highlight-orange); font-weight: bold; }

#player-status-container .level-value { color: var(--text-primary); }

#player-status-container #player-health-numeric { color: var(--text-secondary); font-size: 0.7rem; text-align: center; display: block; }

.teleport-trigger-btn { background: transparent; border: none; font-size: 1.5rem; cursor: pointer; padding: 0; line-height: 1; transition: transform 0.2s, text-shadow 0.2s; }

.teleport-trigger-btn:hover { transform: scale(1.1); text-shadow: 0 0 10px var(--highlight-orange); }

#zone-popup-modal { position: fixed; inset: 0; background-color: rgba(0, 0, 0, 0.7); display: flex; align-items: center; justify-content: center; z-index: 1000; }

#zone-popup-modal.hidden { display: none; }

#zone-list-container li { padding: 0.75rem; background-color: rgba(249, 115, 22, 0.05); border-radius: 0.375rem; cursor: pointer; transition: background-color 0.2s; border: 1px solid rgba(249, 115, 22, 0.1); font-size: 0.9rem; }

#zone-list-container li:hover { background-color: rgba(249, 115, 22, 0.15); border-color: var(--highlight-orange); }

#zone-list-container li.disabled { color: #888; cursor: not-allowed; background-color: rgba(255, 255, 255, 0.02); }

/\* --- PRESERVED COMBAT STATS STYLES --- \*/

#combat-stats-container { padding: 0.5rem; background: rgba(0,0,0,0.2); border-radius: 0.5rem; }

.location-info { text-align: center; margin-bottom: 0.5rem; }

.stats-grid { display: grid; grid-template-columns: 1fr 1fr; gap: 0.5rem; }

.stats-col div { margin-bottom: 2px; }

.stats-label { font-family: Impact, sans-serif; font-size: 12px; font-weight: 700; letter-spacing: 2px; text-transform: uppercase; text-shadow: 1px 1px 2px rgba(0,0,0,0.7); }

.stats-value { font-family: 'Inter', sans-serif; font-weight: 600; color: white; margin-left: 0.5rem; }

.label-location { color: var(--color-text-location); }

.label-health { color: var(--color-text-health); }

.label-exp { color: var(--color-text-exp); }

.label-next-level { color: var(--color-text-next-level); }

.label-drop { color: var(--color-text-drop); }

.label-level { color: var(--color-text-level); }

.label-gold { color: var(--color-text-gold); }

#gold-value { color: var(--color-text-gold); }

.status-ok { color: var(--color-status-ok) !important; }

.status-warn { color: var(--color-status-warn) !important; }

.status-danger { color: var(--color-status-danger) !important; }

/\* --- END PRESERVED STYLES --- \*/

#focus-mode-btn {

position: absolute;

bottom: 8px;

right: 8px;

width: 32px;

height: 32px;

border-radius: 50%;

background-color: var(--glow-red);

border: 1px solid rgba(255, 255, 255, 0.3);

display: flex;

align-items: center;

justify-content: center;

cursor: pointer;

transition: background-color 0.2s;

z-index: 51; /\* Ensures button is above the focused panel \*/

}

#focus-mode-btn:hover { background-color: #ff4d48; }

#focus-mode-btn svg { width: 18px; height: 18px; stroke: white; stroke-width: 2; }

#monsterSelect {

background: var(--input-bg);

border: 1px solid var(--border-color-main);

color: var(--text-primary);

font-family: 'Orbitron', sans-serif;

padding: 0.5rem;

border-radius: 0.375rem;

transition: all 0.2s;

}

#monsterSelect:focus {

outline: none;

border-color: var(--highlight-orange);

box-shadow: 0 0 10px rgba(249, 115, 22, 0.4);

}

#monster-hp-text { color: var(--glow-red); font-weight: bold; text-shadow: 0 0 5px var(--glow-red); }

#enemy-defeated-msg { font-size: 1.5rem; color: var(--glow-red); font-weight: bold; display: none; font-family: 'Orbitron'; }

/\* --- Smart Tooltip / Action Modal --- \*/

#item-action-modal-backdrop {

position: fixed;

inset: 0;

background-color: rgba(0, 0, 0, 0.6);

z-index: 998;

display: none; /\* Hidden by default \*/

}

#item-action-modal-content {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

z-index: 999;

width: 90%;

max-width: 280px;

display: none; /\* Hidden by default \*/

}

.item-action-modal-body {

text-align: center;

}

.item-action-modal-body .item-name {

font-family: 'Orbitron', sans-serif;

font-size: 1.25rem;

margin-bottom: 0.25rem;

}

.item-action-modal-body .item-type {

font-size: 0.8rem;

color: var(--text-secondary);

margin-bottom: 0.75rem;

}

.item-action-modal-body .item-stat {

font-size: 1rem;

margin-bottom: 0.75rem;

}

.item-action-modal-body .item-stat-label {

color: var(--text-secondary);

}

.item-action-modal-body .item-stat-value {

font-weight: 600;

color: var(--text-primary);

}

/\* --- Chat System Styles --- \*/

.footer-tab-button.active { color: var(--highlight-orange); font-weight: bold; }

.footer-chat-input { background: var(--input-bg); border: 1px solid rgba(249, 115, 22, 0.25); color: var(--text-primary); }

.footer-chat-input::placeholder { color: var(--text-secondary); }

.footer-chat-input:focus { outline: none; border-color: var(--highlight-orange); }

.sidebar-closed { transform: translateX(-100%); }

.sidebar-open { transform: translateX(0); }

.chat-bubble { background: rgba(10, 10, 10, 0.7) !important; -webkit-backdrop-filter: blur(10px); backdrop-filter: blur(10px); border: 1px solid; position: relative; }

.chat-bubble-user { border-color: rgba(249, 115, 22, 0.4); }

.chat-bubble-other { border-color: rgba(249, 115, 22, 0.2); }

.reply-icon { cursor: pointer; opacity: 0.4; transition: opacity 0.2s; }

.message-wrapper:hover .reply-icon { opacity: 1; }

.reply-quote { background: rgba(0,0,0,0.3); border-left: 2px solid var(--highlight-orange); padding: 6px 10px; border-radius: 4px; margin-bottom: 8px; font-size: 0.8rem; }

.avatar { width: 40px; height: 40px; border-radius: 50%; object-fit: cover; border: 2px solid rgba(249, 115, 22, 0.25); }

.btn-primary { background-color: transparent; border: 2px solid var(--highlight-orange); color: var(--highlight-orange); transition: all 0.3s ease; }

.btn-primary:hover { background-color: var(--highlight-orange); color: #fff; box-shadow: 0 0 15px var(--highlight-orange); }

#chat-modal .tab { cursor: pointer; transition: all 0.2s; border-bottom: 3px solid transparent; padding-bottom: 8px; color: var(--text-secondary); }

#chat-modal .tab.active { color: var(--highlight-orange); border-bottom-color: var(--highlight-orange); animation: pulse-main-tab-glow 4s ease-in-out infinite; }

#chat-modal input { background: var(--input-bg); border: 1px solid rgba(249, 115, 22, 0.25); border-radius: 0.5rem; padding: 0.75rem 1rem; transition: all 0.2s; color: var(--text-primary); }

#chat-modal input::placeholder { color: var(--text-secondary); }

#chat-modal input:focus { outline: none; border-color: var(--highlight-orange); box-shadow: 0 0 10px rgba(249, 115, 22, 0.4); }

/\* Quest Log Styles \*/

.quest-item { background: rgba(0,0,0,0.2); border: 1px solid rgba(249, 115, 22, 0.15); border-radius: 0.5rem; padding: 1rem; margin-bottom: 0.75rem; }

.quest-title { font-family: 'Orbitron', sans-serif; font-size: 1.1rem; color: var(--highlight-orange); }

.quest-objective { color: var(--text-primary); }

.quest-rewards { color: var(--text-secondary); font-size: 0.9rem; }

.quest-reward-gold { color: var(--color-text-gold); }

.quest-reward-xp { color: var(--color-text-exp); }

.quest-streak-panel { background: rgba(0,0,0,0.3); border: 1px solid var(--border-color-main); padding: 1rem; border-radius: 0.5rem; margin-top: 1rem; text-align: center; }

.quest-streak-value { color: var(--highlight-orange); font-size: 1.5rem; font-family: 'Orbitron'; }

.quest-pool-item { font-size: 0.8rem; }

/\* --- Game Data Editor Styles --- \*/

#game-data-editor-modal { z-index: 100; }

.editor-tab-content { display: none; }

.editor-tab-content.active { display: block; }

.editor-input, .editor-select, .editor-textarea, .editor-checkbox {

background-color: rgba(0,0,0,0.3);

border: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15));

color: var(--highlight-powder-blue, #f0f0f0);

padding: 4px 8px;

border-radius: 4px;

width: 100%;

font-size: 12px;

}

.editor-checkbox { width: auto; }

.editor-input:focus, .editor-select:focus, .editor-textarea:focus {

outline: none;

border-color: var(--glow-vibrant-teal, #f97316);

box-shadow: 0 0 5px var(--glow-vibrant-teal, #f97316);

}

.editor-textarea { min-height: 80px; resize: vertical; font-family: monospace; }

.editor-field-group { margin-bottom: 1rem; border: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15)); padding: 0.75rem; border-radius: 6px; }

.editor-field-group h5 { font-weight: 600; font-size: 0.8rem; margin-bottom: 0.5rem; color: var(--highlight-powder-blue, #f0f0f0); text-transform: uppercase; letter-spacing: 0.05em; border-bottom: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15)); padding-bottom: 0.25rem;}

.editor-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1rem; }

.col-span-full { grid-column: 1 / -1; }

.editor-accordion { border: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15)); border-radius: 8px; margin-bottom: 1rem; background: rgba(0,0,0,0.2); }

.editor-accordion-header { background: rgba(32, 140, 140, 0.1); padding: 0.75rem; cursor: pointer; display: flex; justify-content: space-between; align-items: center; }

.editor-accordion-header:hover { background: rgba(32, 140, 140, 0.2); }

.editor-accordion-header h4 { font-family: 'Orbitron', serif; font-size: 1.25rem; flex-grow: 1; }

.editor-accordion-content { padding: 1rem; display: none; border-top: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15)); }

.editor-accordion.open .editor-accordion-content { display: block; }

.editor-accordion-content label { display: block; margin-bottom: 0.25rem; font-size: 0.75rem; font-weight: 500; color: #9ca3af; }

.dynamic-list-container { border: 1px solid var(--shadow-dark-grey, rgba(249, 115, 22, 0.15)); padding: 0.75rem; border-radius: 6px; margin-top: 1rem; }

.dynamic-list-item { display: grid; grid-template-columns: repeat(3, 1fr) auto; gap: 0.5rem; margin-bottom: 0.5rem; background-color: rgba(0,0,0,0.2); padding: 0.5rem; border-radius: 4px; align-items: center;}

.modal { z-index: 200; }

.loader { width: 48px; height: 48px; border: 5px solid var(--highlight-powder-blue, #f0f0f0); border-bottom-color: transparent; border-radius: 50%; display: inline-block; box-sizing: border-box; animation: rotation 1s linear infinite; }

@keyframes rotation { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }

.infusion-grid {

display: grid;

grid-template-columns: 1fr 1.5fr 1fr; /\* Three columns: Left, Center, Right \*/

gap: 1rem;

height: 100%;

}

.infusion-panel {

background: rgba(0,0,0,0.2);

border: 1px solid rgba(249, 115, 22, 0.15);

border-radius: 0.5rem;

padding: 0.75rem;

display: flex;

flex-direction: column;

min-height: 0;

}

.infusion-panel-title {

font-family: 'Orbitron', sans-serif;

text-align: center;

margin-bottom: 0.75rem;

flex-shrink: 0;

}

.infusion-content-area {

flex-grow: 1;

overflow-y: auto;

}

.infusion-item-entry {

display: flex;

align-items: center;

padding: 0.5rem;

margin-bottom: 0.5rem;

background-color: rgba(0,0,0,0.3);

border: 1px solid transparent;

border-radius: 0.375rem;

cursor: pointer;

transition: all 0.2s;

}

.infusion-item-entry:hover {

border-color: var(--highlight-orange);

background-color: rgba(249, 115, 22, 0.1);

}

.infusion-item-entry img {

width: 40px;

height: 40px;

margin-right: 0.75rem;

object-fit: contain;

}

.infusion-item-info .item-name {

font-weight: 600;

color: var(--text-primary);

}

.infusion-item-info .item-details {

font-size: 0.75rem;

color: var(--text-secondary);

}

.infusion-item-entry.selected {

border-color: var(--glow-red);

box-shadow: 0 0 10px var(--glow-red);

background-color: rgba(239, 68, 68, 0.1);

}

.focused-item-container {

display: flex;

flex-direction: column;

align-items: center;

gap: 1rem;

padding: 1rem;

}

.focused-item-container img {

width: 100px;

height: 100px;

object-fit: contain;

}

.focused-item-details .item-name {

font-family: 'Orbitron', sans-serif;

font-size: 1.25rem;

text-align: center;

}

.focused-item-details .item-tier {

font-size: 0.8rem;

color: var(--text-secondary);

text-align: center;

}

.sockets-container {

display: flex;

gap: 0.75rem;

margin-top: 1rem;

}

.infusion-socket-slot {

width: 52px;

height: 52px;

background-color: rgba(0,0,0,0.5);

border: 2px dashed rgba(249, 115, 22, 0.2);

border-radius: 0.375rem;

display: flex;

align-items: center;

justify-content: center;

cursor: pointer;

transition: all 0.2s;

position: relative;

}

.infusion-socket-slot:hover {

border-color: var(--highlight-orange);

}

.infusion-socket-slot.has-gem {

border-style: solid;

border-color: rgba(249, 115, 22, 0.4);

}

.infusion-socket-slot img {

width: 44px;

height: 44px;

}

.infusion-category {

margin-bottom: 0.5rem;

}

.infusion-category-header {

width: 100%;

padding: 0.6rem 0.5rem;

background-color: rgba(249, 115, 22, 0.05);

border: 1px solid rgba(249, 115, 22, 0.15);

border-radius: 0.375rem;

text-align: left;

font-family: 'Orbitron', sans-serif;

font-size: 0.9rem;

color: var(--text-primary);

cursor: pointer;

transition: background-color 0.2s;

display: flex;

justify-content: space-between;

align-items: center;

}

.infusion-category-header:hover {

background-color: rgba(249, 115, 22, 0.1);

}

.infusion-category-content {

max-height: 0;

overflow: hidden;

transition: max-height 0.3s ease-out;

padding-left: 0.5rem; /\* Indent items \*/

border-left: 2px solid rgba(249, 115, 22, 0.1);

margin-top: 0.25rem;

}

.infusion-category.open .infusion-category-content {

max-height: 1000px; /\* Arbitrary large number \*/

}

.infusion-category-header .arrow {

transition: transform 0.3s ease-out;

}

.infusion-category.open .arrow {

transform: rotate(90deg);

}

#infusion-gem-pouch-content {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(52px, 1fr));

gap: 0.5rem;

}

.infusion-gem-item {

position: relative;

border: 2px solid transparent;

border-radius: 0.375rem;

background: transparent;

cursor: pointer;

aspect-ratio: 1 / 1;

display: flex;

align-items: center;

justify-content: center;

transition: all 0.2s;

}

.infusion-gem-item:hover {

border-color: var(--highlight-orange);

}

.infusion-gem-item.selected {

border-color: var(--glow-red);

box-shadow: 0 0 10px var(--glow-red);

}

.infusion-gem-category-content {

padding-top: 0.75rem;

}

.gem-filter-bar {

width: 100%;

padding: 0.5rem;

margin-bottom: 0.5rem;

background-color: rgba(0,0,0,0.5);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

text-align: center;

font-family: 'Orbitron', sans-serif;

color: var(--text-primary);

cursor: pointer;

transition: all 0.2s;

position: relative;

}

.gem-filter-bar:hover {

border-color: var(--highlight-orange);

background-color: rgba(249, 115, 22, 0.1);

}

.gem-filter-dropdown {

position: absolute;

width: 100%;

background-color: var(--panel-bg);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

z-index: 10;

max-height: 200px;

overflow-y: auto;

backdrop-filter: blur(5px);

}

.gem-filter-option {

padding: 0.6rem;

cursor: pointer;

transition: background-color 0.2s;

text-align: center;

font-size: 0.9rem;

}

.gem-filter-option:hover {

background-color: rgba(249, 115, 22, 0.15);

}

.infusion-panel {

background: rgba(0,0,0,0.2);

border: 1px solid rgba(249, 115, 22, 0.15);

border-radius: 0.5rem;

padding: 0.75rem;

display: flex;

flex-direction: column;

min-height: 0; /\* <-- IMPORTANT: This fixes the layout bug \*/

}

.gem-filter-bar {

width: 100%;

padding: 0.5rem;

margin-bottom: 0.5rem;

background-color: rgba(0,0,0,0.5);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

text-align: center;

font-family: 'Orbitron', sans-serif;

color: var(--text-primary);

cursor: pointer;

transition: all 0.2s;

position: relative;

}

.gem-filter-bar:hover {

border-color: var(--highlight-orange);

background-color: rgba(249, 115, 22, 0.1);

}

.gem-filter-dropdown {

position: absolute;

width: 100%;

background-color: var(--panel-bg);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

z-index: 10;

max-height: 200px;

overflow-y: auto;

backdrop-filter: blur(5px);

}

.gem-filter-option {

padding: 0.6rem;

cursor: pointer;

transition: background-color 0.2s;

text-align: center;

font-size: 0.9rem;

}

.gem-filter-option:hover {

background-color: rgba(249, 115, 22, 0.15);

}

.infusion-panel {

background: rgba(0,0,0,0.2);

border: 1px solid rgba(249, 115, 22, 0.15);

border-radius: 0.5rem;

padding: 0.75rem;

display: flex;

flex-direction: column;

min-height: 0; /\* <-- IMPORTANT: This fixes the layout bug \*/

}

.gem-filter-bar {

width: 100%;

padding: 0.5rem;

margin-bottom: 0.5rem;

background-color: rgba(0,0,0,0.5);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

text-align: center;

font-family: 'Orbitron', sans-serif;

color: var(--text-primary);

cursor: pointer;

transition: all 0.2s;

position: relative;

}

.gem-filter-bar:hover {

border-color: var(--highlight-orange);

background-color: rgba(249, 115, 22, 0.1);

}

.gem-filter-dropdown {

position: absolute;

width: 100%;

background-color: var(--panel-bg);

border: 1px solid var(--border-color-main);

border-radius: 0.375rem;

z-index: 10;

max-height: 200px;

overflow-y: auto;

backdrop-filter: blur(5px);

}

.gem-filter-option {

padding: 0.6rem;

cursor: pointer;

transition: background-color 0.2s;

text-align: center;

font-size: 0.9rem;

}

.gem-filter-option:hover {

background-color: rgba(249, 115, 22, 0.15);

}

.gem-dot-container {

position: absolute;

top: 3px;

left: 3px;

display: flex;

gap: 3px;

pointer-events: none; /\* So they don't interfere with clicks \*/

}

.gem-dot {

width: 6px;

height: 6px;

background-color: var(--highlight-orange);

border-radius: 50%;

border: 1px solid rgba(0,0,0,0.7);

box-shadow: 0 0 4px var(--highlight-orange);

}

.item-gem-list {

margin-top: 0.75rem;

border-top: 1px solid rgba(249, 115, 22, 0.2);

padding-top: 0.75rem;

display: flex;

flex-direction: column;

gap: 0.5rem;

}

.item-gem-entry {

display: flex;

justify-content: space-between;

align-items: center;

font-size: 0.8rem;

color: var(--text-secondary);

}

.item-gem-name {

font-weight: 600;

color: var(--text-primary);

}

.item-gem-effect {

color: var(--highlight-orange);

font-family: 'Orbitron', sans-serif;

}

.equipment-gem-list {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: 0.7rem;

line-height: 1.1;

font-family: 'Orbitron', sans-serif;

color: var(--text-secondary);

margin-right: 0.5rem; /\* Adds a little space between text and icon \*/

}

</style>

</head>

<body class="bg-black">

<canvas id="smoke-canvas"></canvas>

<div id="game-container" class="h-full">

<div id="game-hud-screen" class="relative z-10 h-full">

<div class="max-w-md mx-auto h-full flex flex-col p-2 gap-2">

<header id="game-section" class="glass-panel p-3 rounded-lg flex justify-between items-center flex-shrink-0">

<section id="player-status-panel" class="flex-1">

<div id="player-status-container" class="flex flex-col items-start space-y-2 flex-grow">

<div class="flex items-center gap-4">

<div>

<span class="level-label font-orbitron">Level: </span>

<span id="player-level-value" class="level-value font-orbitron">1</span>

<span id="player-name-value" class="name-value ml-2 font-orbitron">Player</span>

</div>

<div id="transport-controls">

<button id="zone-teleport-trigger" class="teleport-trigger-btn" title="Teleport to Zone">🌀</button>

</div>

</div>

<div class="health-bar-container w-full max-w-[150px]">

<div class="progress-bar-track h-3"><div id="hp-bar" class="progress-bar-fill h-full" style="width: 100%; background-color: var(--hp-color);"></div></div>

<span id="player-health-numeric">100 / 100</span>

</div>

</div>

</section>

<section id="navigation-panel" class="flex-1 flex justify-end">

<div class="flex items-center gap-2">

<div id="mini-map-container" class="relative w-24 h-24" title="World Map">

<div class="absolute -inset-1 rounded-full border border-dashed border-orange-500/30 animate-spin" style="animation-duration: 20s; animation-timing-function: linear;"></div>

<div class="relative w-full h-full rounded-full overflow-hidden glass-panel border-2 border-[var(--border-color-main)]">

<canvas id="mini-map-canvas"></canvas>

</div>

</div>

<div class="flex flex-col items-center gap-1">

<div id="d-pad-controls">

<div class="game-key move-key" id="key-up" data-key="up"><svg viewBox="0 0 24 24"><path d="M7.41 15.41L12 10.83l4.59 4.58L18 14l-6-6-6 6z"></path></svg></div>

<div class="game-key move-key" id="key-left" data-key="left"><svg viewBox="0 0 24 24"><path d="M15.41 16.59L10.83 12l4.58-4.59L14 6l-6 6 6 6 1.41-1.41z"></path></svg></div>

<div class="game-key move-key" id="key-down" data-key="down"><svg viewBox="0 0 24 24"><path d="M7.41 8.59L12 13.17l4.59-4.58L18 10l-6 6-6-6z"></path></svg></div>

<div class="game-key move-key" id="key-right" data-key="right"><svg viewBox="0 0 24 24"><path d="M8.59 16.59L13.17 12 8.59 7.41 10 6l6 6-6 6-1.41-1.41z"></path></svg></div>

</div>

<div class="game-key" id="key-interact" data-key="interact">Interact</div>

</div>

</div>

</section>

</header>

<section id="main-content-panel" class="flex-grow flex flex-col overflow-hidden">

<main id="main-content" class="flex-grow flex flex-col overflow-hidden glass-panel rounded-lg relative">

<div id="main-tabs-container" class="flex-shrink-0 flex items-center overflow-x-auto whitespace-nowrap custom-scrollbar-x border-b border-[var(--border-color-main)]">

<button class="main-tab-button active" data-tab="equipment">Equipment</button>

<button class="main-tab-button" data-tab="infusion">Infusion</button>

<button class="main-tab-button" data-tab="inventory">Inventory</button>

<button class="main-tab-button" data-tab="stats">Stats</button>

<button class="main-tab-button" data-tab="combat">Combat</button>

<button class="main-tab-button" data-tab="quest">Quest</button>

<button class="main-tab-button" data-tab="settings">Settings</button>

</div>

<div id="main-tab-content" class="flex-grow p-2 md:p-4 overflow-y-auto custom-scrollbar relative">

<div id="tab-content-equipment" class="main-tab-panel active"></div>

<div id="tab-content-infusion" class="main-tab-panel"></div>

<div id="tab-content-inventory" class="main-tab-panel"></div>

<div id="tab-content-stats" class="main-tab-panel"></div>

<div id="tab-content-combat" class="main-tab-panel"></div>

<div id="tab-content-quest" class="main-tab-panel"></div>

<div id="tab-content-settings" class="main-tab-panel"></div>

</div>

<button id="focus-mode-btn" title="Toggle Focus Mode">

<svg id="focus-icon-expand" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" d="M4 8V4m0 0h4M4 4l5 5m11-1V4m0 0h-4m4 0l-5 5M4 16v4m0 0h4m-4 0l5-5m11 5v-4m0 0h-4m4 0l-5-5"></path></svg>

<svg id="focus-icon-collapse" class="hidden" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" d="M15 19l-7-7 7-7"></path></svg>

</button>

</main>

</section>

<section id="communication-panel" class="flex-shrink-0">

<div id="footer-chat-container" class="glass-panel w-full p-2 rounded-lg flex flex-col">

<div class="flex-shrink-0 flex flex-wrap gap-1 mb-2">

<button data-channel="main" class="footer-tab-button glass-button text-xs px-3 py-1 rounded-md flex-grow active">Main</button>

<button data-channel="sales" class="footer-tab-button glass-button text-xs px-3 py-1 rounded-md flex-grow">Sales</button>

<button data-channel="clan" class="footer-tab-button glass-button text-xs px-3 py-1 rounded-md flex-grow">Clan</button>

<button id="open-chat-modal-btn" class="glass-button text-xs px-2 py-1 rounded-md" title="Open Full Chat">

<svg class="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 8V4m0 0h4M4 4l5 5m11-1V4m0 0h-4m4 0l-5 5M4 16v4m0 0h4m-4 0l5-5m11 5v-4m0 0h-4m4 0l-5-5"></path></svg>

</button>

</div>

<div id="footer-chat-content-wrapper" class="text-xs space-y-1 overflow-y-auto custom-scrollbar flex-grow" style="height: 70px;"></div>

<form id="footer-message-form" class="flex-shrink-0 flex gap-2 mt-2">

<input type="text" id="footer-message-input" class="footer-chat-input flex-grow w-full px-2 py-1 text-xs rounded-md" placeholder="Type a message..." autocomplete="off">

<button type="submit" id="footer-send-button" class="glass-button text-xs px-3 py-1 rounded-md">Send</button>

</form>

</div>

</section>

</div>

</div>

</div>

<div id="zone-popup-modal" class="hidden">

<div id="zone-popup-backdrop" class="fixed inset-0"></div>

<div class="zone-popup-content glass-panel p-4 rounded-lg w-11/12 max-w-sm relative">

<div class="flex justify-between items-center mb-4">

<h2 class="font-orbitron text-xl">Teleport to Zone</h2>

<button id="zone-popup-close" class="text-2xl leading-none transition-colors hover:text-[var(--highlight-orange)]">&times;</button>

</div>

<ul id="zone-list-container" class="space-y-1 max-h-80 overflow-y-auto custom-scrollbar">

</ul>

</div>

</div>

<div id="item-action-modal-backdrop"></div>

<div id="item-action-modal-content" class="glass-panel p-4 rounded-lg">

<div id="item-action-modal-body" class="item-action-modal-body">

</div>

</div>

<div id="stat-info-modal">

<div id="stat-info-backdrop"></div>

<div class="stat-info-content">

<h4 id="stat-info-title" class="font-orbitron text-lg text-white mb-2"></h4>

<p id="stat-info-description" class="text-gray-300"></p>

</div>

</div>

<div id="modal-container"></div>

<div id="toast-notification"></div>

<div id="chat-modal" class="modal-backdrop hidden">

<div class="relative w-11/12 max-w-5xl h-[90vh] max-h-[850px] rounded-2xl flex glass-panel overflow-hidden">

<button id="close-chat-modal-btn" class="absolute top-3 right-4 text-gray-400 hover:text-white z-50">

<svg class="w-7 h-7" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12"></path></svg>

</button>

<div id="sidebar-overlay" class="fixed inset-0 bg-black/60 z-30 hidden md:hidden"></div>

<div id="sidebar" class="absolute md:relative z-40 h-full w-4/5 max-w-xs md:w-1/3 md:max-w-[320px] flex flex-col p-4 transition-transform duration-300 ease-in-out sidebar-closed md:sidebar-open glass-panel md:bg-transparent md:border-r md:border-l-0 md:border-t-0 md:border-b-0 md:shadow-none border-white/10">

<button id="close-sidebar-btn" class="md:hidden absolute top-4 right-4 text-gray-300 hover:text-white"><svg class="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12"></path></svg></button>

<div class="mt-8 md:mt-0 p-3 glass-panel rounded-lg">

<div class="flex items-center gap-3">

<img id="sidebar-avatar" src="https://placehold.co/64x64/1a1a1a/f97316?text=J" class="avatar w-12 h-12">

<div class="text-left overflow-hidden">

<p id="sidebar-username" class="font-bold text-lg truncate font-orbitron">JuugBoyTV</p>

<p class="text-xs capitalize" style="opacity: 0.8;">Player</p>

</div>

</div>

</div>

<div class="flex-grow flex flex-col min-h-0 pt-4">

<h2 id="online-users-header" class="text-lg font-orbitron mb-2 pl-2">Online</h2>

<div id="online-users-list" class="flex-grow overflow-y-auto custom-scrollbar pr-2"></div>

</div>

</div>

<div class="flex-1 flex flex-col w-full md:w-auto min-w-0">

<div class="p-4 border-b border-white/10 flex items-center justify-between relative bg-black/20">

<button id="open-sidebar-btn" class="md:hidden"><svg class="w-7 h-7" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16"></path></svg></button>

<div id="tabs-container" class="flex-grow flex justify-center gap-4 md:gap-8 text-md font-orbitron font-bold">

<div class="tab active" data-channel="main">Main Chat</div>

<div class="tab" data-channel="sales">Sales Chat</div>

<div class="tab" data-channel="clan">Clan Chat</div>

</div>

<div class="w-7 md:hidden"></div>

</div>

<div id="content-container" class="flex-grow flex flex-col min-h-0 bg-black/10">

<div id="chat-messages" class="flex-grow p-4 overflow-y-auto custom-scrollbar"></div>

</div>

<div id="typing-indicator" class="px-4 pb-2 text-sm text-gray-500 h-6"></div>

<div id="reply-indicator" class="px-4 pt-2 hidden"><div class="glass-panel bg-opacity-80 rounded-t-lg p-2 text-sm"><div class="flex justify-between items-center"><div><p class="font-semibold" style="color: var(--highlight-orange);">Replying to <span id="reply-username"></span></p><p id="reply-text" class="text-gray-300 truncate"></p></div><button id="cancel-reply-btn" class="text-gray-400 hover:text-white text-2xl">&times;</button></div></div></div>

<form id="message-form" class="p-4 flex items-center gap-3 border-t border-white/10 bg-black/20">

<input type="text" id="message-input" placeholder="Type your message..." autocomplete="off" class="flex-grow">

<button type="submit" id="send-button" class="font-bold py-3 px-5 rounded-lg btn-primary">Send</button>

</form>

</div>

</div>

</div>

<div id="dev-tools-container">

<div id="game-data-editor-modal" class="fixed inset-0 bg-black/80 flex-col p-2 sm:p-4 hidden">

<div class="w-full h-full glass-panel rounded-lg flex flex-col">

<div class="flex-shrink-0 flex flex-wrap justify-between items-center gap-2 p-3 border-b border-[var(--border-color-main)]">

<h2 class="font-orbitron text-xl sm:text-2xl">Game Data Editor</h2>

<div id="editor-status" class="text-sm text-yellow-400 order-last w-full text-center sm:order-none sm:mx-4 sm:w-auto"></div>

<div class="flex items-center gap-2 flex-wrap justify-center sm:justify-end">

<button id="gdd-editor-save-btn" class="glass-button px-3 py-1 sm:px-4 sm:py-2 rounded-md text-xs sm:text-sm">Save</button>

<button id="gdd-editor-reset-btn" class="glass-button px-3 py-1 sm:px-4 sm:py-2 rounded-md text-xs sm:text-sm bg-red-900/50 border-red-500/80">Reset</button>

<button id="gdd-editor-close-btn" class="text-3xl leading-none">&times;</button>

</div>

</div>

<div class="flex flex-1 min-h-0">

<div class="w-40 sm:w-48 flex-shrink-0 p-2 border-r border-[var(--border-color-main)] overflow-y-auto custom-scrollbar">

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="dashboard">Dashboard</button>

<h4 class="font-orbitron text-sm text-center my-2 text-cyan-300">Core Game</h4>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="constants">Formulas</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="levels">XP Curve</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="races">Races</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="zones">Zones</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="monsters">Monsters</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="base\_items">Base Items</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="gear\_tiers">Gear Tiers</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="gems">Gems</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="enchantments">Enchantments</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="quests">Quests</button>

<button class="editor-tab-btn glass-button w-full justify-start text-sm px-3 py-2 mb-1" data-tab="loot\_tables">Loot Tables</button>

</div>

<div id="gdd-editor-content" class="flex-grow p-2 sm:p-4 overflow-y-auto custom-scrollbar">

</div>

</div>

</div>

</div>

<div id="picker-modal" class="hidden modal fixed inset-0 bg-black/80 flex items-center justify-center" style="z-index: 210;"></div>

<div id="confirmation-modal" class="hidden modal fixed inset-0 bg-black/80 flex items-center justify-center"></div>

<div id="prompt-modal" class="hidden modal fixed inset-0 bg-black/80 flex items-center justify-center"></div>

<div id="loading-modal" class="hidden modal fixed inset-0 bg-black/80 flex items-center justify-center" style="z-index: 250;"></div>

</div>

<script type="module">

// --- App State & Config ---

let state = {

player: {},

ui: { isFocused: false, selectedInventoryId: null, selectedGemId: null },

game: { combatActive: false, currentZoneTier: 1, globalJackpot: 0 },

keyState: { up: false, left: false, down: false, right: false, interact: false },

};

// --- UI Elements ---

const ui = {};

document.querySelectorAll('[id]').forEach(el => {

const camelCaseId = el.id.replace(/-(\w)/g, (m, g) => g.toUpperCase());

ui[camelCaseId] = el;

});

// --- Utility Functions ---

function showToast(message, isError = false) {

ui.toastNotification.textContent = message;

ui.toastNotification.className = `glass-panel fixed left-1/2 -translate-x-1/2 z-[210] transition-all duration-500 ease-in-out px-6 py-3 rounded-lg font-semibold ${isError ? 'toast-error' : 'toast-success'}`;

ui.toastNotification.style.bottom = '5rem';

setTimeout(() => { ui.toastNotification.style.bottom = '-100px'; }, 3000);

}

function logToGame(message) {

const combatLog = document.getElementById('combat-log');

if (combatLog) {

combatLog.innerHTML = `<strong class="text-[var(--highlight-orange)]">Log:</strong> ${message}`;

}

console.log(`Game Log: ${message}`);

}

// --- Background Animations ---

const smokeCanvas = document.getElementById('smoke-canvas');

const smokeCtx = smokeCanvas.getContext('2d');

let smokeParticles = [];

const resizeSmokeCanvas = () => {

smokeCanvas.width = window.innerWidth;

smokeCanvas.height = window.innerHeight;

smokeParticles = Array.from({ length: 75 }, () => ({

x: Math.random() \* smokeCanvas.width, y: Math.random() \* smokeCanvas.height,

size: Math.random() \* 150 + 50,

speedX: Math.random() \* 0.4 - 0.2, speedY: Math.random() \* 0.4 - 0.2,

color: `rgba(249, 115, 22, ${Math.random() \* 0.07})`

}));

};

const animateSmoke = () => {

smokeCtx.clearRect(0, 0, smokeCanvas.width, smokeCanvas.height);

smokeParticles.forEach(p => {

p.x += p.speedX; p.y += p.speedY;

if (p.x < -p.size) p.x = smokeCanvas.width + p.size; if (p.x > smokeCanvas.width + p.size) p.x = -p.size;

if (p.y < -p.size) p.y = smokeCanvas.height + p.size; if (p.y > smokeCanvas.height + p.size) p.y = -p.size;

smokeCtx.fillStyle = p.color; smokeCtx.beginPath(); smokeCtx.arc(p.x, p.y, p.size, 0, Math.PI \* 2);

smokeCtx.filter = 'blur(60px)'; smokeCtx.fill();

});

requestAnimationFrame(animateSmoke);

};

window.addEventListener('resize', resizeSmokeCanvas);

resizeSmokeCanvas();

animateSmoke();

// --- NEW: SHARED UTILITIES ---

/\*\*

\* HexUtils - A shared utility object for hexagonal grid mathematics.

\* Handles all coordinate conversions and neighbor calculations.

\*/

const HexUtils = {

/\*\*

\* Converts axial hex coordinates to pixel coordinates (pointy-top).

\* @param {number} q - The q coordinate of the hex.

\* @param {number} r - The r coordinate of the hex.

\* @param {number} size - The radius of the hex.

\* @returns {{x: number, y: number}} The pixel coordinates.

\*/

hexToPixel: (q, r, size) => {

const x = size \* (3/2 \* q);

const y = size \* (Math.sqrt(3)/2 \* q + Math.sqrt(3) \* r);

return {x, y};

},

/\*\*

\* Converts pixel coordinates to axial hex coordinates.

\* @param {number} x - The x pixel coordinate.

\* @param {number} y - The y pixel coordinate.

\* @param {number} size - The radius of the hex.

\* @returns {{q: number, r: number}} The rounded hex coordinates.

\*/

pixelToHex(x, y, size) {

const q = (2/3 \* x) / size;

const r = (-1/3 \* x + Math.sqrt(3)/3 \* y) / size;

return this.hexRound(q, r);

},

/\*\*

\* Rounds fractional hex coordinates to the nearest integer hex coordinates.

\* @param {number} fq - The fractional q coordinate.

\* @param {number} fr - The fractional r coordinate.

\* @returns {{q: number, r: number}} The integer hex coordinates.

\*/

hexRound(fq, fr) {

const fs = -fq - fr;

let q = Math.round(fq);

let r = Math.round(fr);

let s = Math.round(fs);

const q\_d = Math.abs(q - fq);

const r\_d = Math.abs(r - fr);

const s\_d = Math.abs(s - fs);

if (q\_d > r\_d && q\_d > s\_d) {

q = -r - s;

} else if (r\_d > s\_d) {

r = -q - s;

}

return { q, r };

},

/\*\*

\* Gets the coordinates of all 6 neighbors for a given hex.

\* @param {number} q - The q coordinate of the hex.

\* @param {number} r - The r coordinate of the hex.

\* @returns {Array<{q: number, r: number}>} An array of neighbor coordinates.

\*/

getNeighbors: (q, r) => [

{ q: 1, r: 0 }, { q: 1, r: -1 }, { q: 0, r: -1 },

{ q: -1, r: 0 }, { q: -1, r: 1 }, { q: 0, r: 1 }

].map(dir => ({ q: q + dir.q, r: r + dir.r })),

};

// --- GDD GAME DATA & ITEM FACTORY ---

let AllZones = {}; // This will be populated by loadGameData

const GameData = {

ItemFactory: {

baseItemTemplates: [

// Armor & Apparel - All sockets set to 2

{ id: 'base\_helm\_1', name: 'Iron Helm', type: 'Helmet', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1396.png', proportion: 0.75, sockets: 2 },

{ id: 'base\_armor\_1', name: 'Steel Plate Armor', type: 'Armor', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1401.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_leggings\_1', name: 'Steel Greaves', type: 'Leggings', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1402.png', proportion: 0.5, sockets: 2, special: { hitChanceBonus: 0.10 } },

{ id: 'base\_boots\_1', name: 'Steel Sabatons', type: 'Boots', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1403.png', proportion: 0.75, sockets: 2 },

{ id: 'base\_gauntlets\_1', name: 'Steel Gauntlets', type: 'Gauntlets', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1404.png', proportion: 0.5, sockets: 2, special: { classBonus: 0.15 } },

{ id: 'base\_necklace\_1', name: 'Amulet of Power', type: 'Amulet', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1405.png', proportion: 0.0, sockets: 2 },

{ id: 'base\_ring\_1', name: 'Ring of Vitality', type: 'Ring', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/juugboytv-equipment1/IMG\_1406.png', proportion: 0.0, sockets: 2 },

// Weapons - SubTypes added and sockets set to 2

{ id: 'base\_shield\_1', name: 'Heater Shield', type: 'Weapon', subType: 'Shield', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1408.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_coh\_1', name: 'Caster Off-Hand', type: 'Weapon', subType: 'Caster Off-Hand', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1409.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_bow\_1', name: 'Longbow', type: 'Weapon', subType: 'Bow', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1410.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_axe\_1', name: 'Battle Axe', type: 'Weapon', subType: 'Axe', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1411.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_sword\_1', name: 'Knightly Sword', type: 'Weapon', subType: 'Sword', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/Weapons/IMG\_1412.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_dagger\_1', name: 'Rondel Dagger', type: 'Weapon', subType: 'Dagger', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1413.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_mace\_1', name: 'Flanged Mace', type: 'Weapon', subType: 'Mace', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1414.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_cstaff\_1', name: 'Caster Staff', type: 'Weapon', subType: 'Caster Staff', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1415.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_fstaff\_1', name: 'Fighter Staff', type: 'Weapon', subType: 'Fighter Staff', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1416.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_claws\_1', name: 'Claws', type: 'Weapon', subType: 'Claws', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Weapons/IMG\_1417.png', proportion: 1.0, sockets: 2 },

// Spellbooks - SubTypes added and sockets set to 2

{ id: 'base\_airspell\_1', name: 'Air Spell', type: 'Spellbook', subType: 'Air', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1425.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_deathspell\_1', name: 'Death Spell', type: 'Spellbook', subType: 'Death', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1424.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_coldspell\_1', name: 'Cold Spell', type: 'Spellbook', subType: 'Cold', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1423.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_firespell\_1', name: 'Fire Spell', type: 'Spellbook', subType: 'Fire', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1422.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_arcanespell\_1', name: 'Arcane Spell', type: 'Spellbook', subType: 'Arcane', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1421.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_earthspell\_1', name: 'Earth Spell', type: 'Spellbook', subType: 'Earth', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_1420.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_fighterbuff\_1', name: 'Fighter Buff Spell', type: 'Spellbook', subType: 'Fighter Buff', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_2292.png', proportion: 1.0, sockets: 2 },

{ id: 'base\_drainspell\_1', name: 'Drain Spell', type: 'Spellbook', subType: 'Drain', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Spells/IMG\_2358.png', proportion: 1.0, sockets: 2 },

],

createItemInstance(baseItemId, tier, type, options = {}) {

const baseItem = this.baseItemTemplates.find(b => b.id === baseItemId);

if (!baseItem) {

console.error(`Base item with ID "${baseItemId}" not found.`);

return null;

}

const baseClassValue = 13 \* Math.pow(1.22, tier - 1);

const newItem = {

instanceId: `inst\_${type[0]}\_${baseItem.id}\_${Date.now()}\_${Math.random()}`,

baseItemId: baseItem.id,

type: type,

tier: tier,

socketedGems: [],

special: baseItem.special || null,

};

if (type === 'Shadow') {

newItem.quality = options.quality || (0.75 + (Math.random() \* 0.75));

newItem.kills = 0; // Initialize kills for progression

} else if (type === 'Echo') {

newItem.quality = options.quality;

}

// Recalculate classValue based on quality

let finalClassValue = baseClassValue \* baseItem.proportion;

if (newItem.quality) {

finalClassValue \*= newItem.quality;

}

newItem.classValue = finalClassValue;

return newItem;

}

},

Gems: {

lorestone: { name: 'LoreStone', abbreviation: 'LST', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1500.png', effect: "Increase Base Spell Class", values: [1.5, 2.5, 3.5, 5, 7, 9, 11, 13, 15] },

loreheart: { name: 'LoreHeart', abbreviation: 'LHT', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1501.png', effect: "Increase Base SC & AC", values: { sc: [1, 1.5, 2.5, 3.5, 5, 7, 8.5, 10, 12], ac: [1.5, 2.5, 3.5, 5, 7, 9, 11, 13, 15] } },

mindrite: { name: 'Mindrite', abbreviation: 'MDR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1504.png', effect: "Increase Wisdom", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

dullrite: { name: 'Dullrite', abbreviation: 'DLR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1505.png', effect: "Decrease Enemy Wisdom", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

drainrite: { name: 'Drainrite', abbreviation: 'DRR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1506.png', effect: "Steal Enemy Wisdom", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

mindstone: { name: 'MindStone', abbreviation: 'MDS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1507.png', effect: "Increase Intelligence", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

dullstone: { name: 'DullStone', abbreviation: 'DLS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1508.png', effect: "Decrease Enemy Intelligence", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

drawstone: { name: 'DrawStone', abbreviation: 'DRS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1509.png', effect: "Steal Enemy Intelligence", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

sagerite: { name: 'Sagerite', abbreviation: 'SGR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1510.png', effect: "Increase Ntl & Wisdom", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

drowseite: { name: 'Drowseite', abbreviation: 'DWS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1511.png', effect: "Decrease Enemy Ntl & Wisdom", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

leechrite: { name: 'Leechrite', abbreviation: 'LCR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1512.png', effect: "Steal Enemy Ntl & Wisdom", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

warstone: { name: 'WarStone', abbreviation: 'WST', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1503.png', effect: "Increase Base Weapon Class", values: [1.5, 2.5, 3.5, 5, 7, 9, 11, 13, 15] },

warheart: { name: 'WarHeart', abbreviation: 'WHT', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1502.png', effect: "Increase Base WC & AC", values: { wc: [1, 1.5, 2.5, 3.5, 5, 7, 8.5, 10, 12], ac: [1.5, 2.5, 3.5, 5, 7, 9, 11, 13, 15] } },

agilite: { name: 'Agilite', abbreviation: 'AGL', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1513.png', effect: "Increase Dexterity", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

cripplite: { name: 'Cripplite', abbreviation: 'CPL', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1514.png', effect: "Decrease Enemy Dexterity", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

siphilite: { name: 'Siphilite', abbreviation: 'SPH', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1516.png', effect: "Steal Enemy Dexterity", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

mightstone: { name: 'MightStone', abbreviation: 'MGS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1517.png', effect: "Increase Strength", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

weakstone: { name: 'WeakStone', abbreviation: 'WKS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1525.png', effect: "Decrease Enemy Strength", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

sapstone: { name: 'SapStone', abbreviation: 'SPS', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1524.png', effect: "Steal Enemy Strength", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

vigorite: { name: 'Vigorite', abbreviation: 'VGR', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1523.png', effect: "Increase STR & DEX", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

debilitate: { name: 'Debilitate', abbreviation: 'DBT', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1522.png', effect: "Decrease Enemy STR & DEX", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

syphonite: { name: 'Syphonite', abbreviation: 'SYN', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gems/IMG\_1521.png', effect: "Steal Enemy STR & DEX", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

obsidian\_heart: { name: 'Obsidian Heart', abbreviation: 'OH', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1540.png', effect: "Increase Base Armor Class", values: [1.5, 2.5, 3.5, 5, 7, 9, 11, 13, 15] },

spike\_core: { name: 'Spike-Core', abbreviation: 'SPC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1539.png', effect: "Increase Critical Hit Chance", values: [1, 2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20] },

true\_core: { name: 'True-Core', abbreviation: 'TRC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1538.png', effect: "Increase Hit Chance", values: [3, 6, 9, 12, 15, 18, 21, 25, 30] },

veil\_core: { name: 'Veil-Core', abbreviation: 'VLC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1534.png', effect: "Decrease Enemy Hit Chance", values: [2.7, 5.4, 8.1, 10.8, 13.5, 16.2, 18.9, 22.5, 27] },

vital\_core: { name: 'Vital-Core', abbreviation: 'VTC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1533.png', effect: "Heals", values: [5, 7.5, 10, 12.5, 15, 20, 30, 40, 50] },

blood\_core: { name: 'Blood-Core', abbreviation: 'BDC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1532.png', effect: "Steal Enemy Health", values: [4, 6, 9, 15, 25, 40, 50, 60, 75] },

flame\_core: { name: 'Flame-Core', abbreviation: 'FLC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1530.png', effect: "Damages Enemy", values: [8, 10, 12, 14, 16, 19, 22, 26, 30] },

treasure\_core: { name: 'Treasure-Core', abbreviation: 'TRC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1529.png', effect: "Increase Drop Chance", values: [2, 3, 4, 5, 6, 7, 8, 9, 10] },

ascend\_core: { name: 'Ascend-Core', abbreviation: 'ASC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1528.png', effect: "Increase Experience Gain", values: [2, 4, 6, 9, 12, 15, 19, 24, 30] },

midas\_core: { name: 'Midas-Core', abbreviation: 'MDC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1526.png', effect: "Increase Gold Earned", values: [2, 4, 6, 9, 12, 15, 19, 24, 30] },

masterwork\_core: { name: 'Masterwork-Core', abbreviation: 'MWC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1527.png', effect: "Increase Mastery Chance", values: [5, 10, 15, 20, 25, 30, 35, 40, 50] },

echoing\_core: { name: 'Echoing-Core', abbreviation: 'ECC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1520.png', effect: "Increase Double Hit Chance", values: [2, 3, 4, 5, 6, 7, 8, 10, 12] },

harvester\_core: { name: 'Harvester-Core', abbreviation: 'HVC', imageUrl: 'https://raw.githubusercontent.com/juugboytv/Geminus/refs/heads/Gemsmisc/IMG\_1542.png', effect: "Increase Resource Drop Chance", values: [5, 10, 15, 20, 30, 50, 55, 60, 75] },

},

GemGradeUnlockLevels: [1, 100, 253, 1000, 6143, 13636, 35452, 83333, 172222],

Races: { 'Dwarf': { class: 'Fighter', stats: { STR: 12, DEX: 8, VIT: 10, NTL: 12, WIS: 8 } }, 'Elf': { class: 'Caster', stats: { STR: 6, DEX: 14, VIT: 6, NTL: 12, WIS: 12 } }, 'Halfling': { class: 'Fighter', stats: { STR: 4, DEX: 18, VIT: 8, NTL: 2, WIS: 8 } }, 'Human': { class: 'Fighter', stats: { STR: 8, DEX: 14, VIT: 8, NTL: 5, WIS: 5 } }, 'Gnome': { class: 'Caster', stats: { STR: 2, DEX: 2, VIT: 6, NTL: 12, WIS: 18 } }, 'Dragonborn': { class: 'Fighter', stats: { STR: 18, DEX: 8, VIT: 8, NTL: 9, WIS: 7 } }, 'Tiefling': { class: 'Caster', stats: { STR: 2, DEX: 6, VIT: 6, NTL: 18, WIS: 8 } }, 'Hobbit': { class: 'Fighter', stats: { STR: 4, DEX: 20, VIT: 12, NTL: 2, WIS: 2 } }, 'Orc': { class: 'Fighter', stats: { STR: 18, DEX: 6, VIT: 12, NTL: 2, WIS: 2 } }, 'Troll': { class: 'Fighter', stats: { STR: 14, DEX: 8, VIT: 14, NTL: 2, WIS: 2 } }, 'Minotaur': { class: 'Fighter', stats: { STR: 16, DEX: 8, VIT: 8, NTL: 2, WIS: 2 } }, 'Centaur': { class: 'Fighter', stats: { STR: 12, DEX: 16, VIT: 8, NTL: 2, WIS: 2 } }, 'Griffin': { class: 'Caster', stats: { STR: 4, DEX: 4, VIT: 8, NTL: 12, WIS: 12 } }, 'Phoenix': { class: 'Caster', stats: { STR: 2, DEX: 4, VIT: 6, NTL: 20, WIS: 8 } }, 'Unicorn': { class: 'Caster', stats: { STR: 2, DEX: 4, VIT: 6, NTL: 12, WIS: 16 } }, 'Baba Yaga': { class: 'Caster', stats: { STR: 2, DEX: 2, VIT: 6, NTL: 18, WIS: 12 } }, 'Draugr': { class: 'Caster', stats: { STR: 4, DEX: 4, VIT: 4, NTL: 16, WIS: 12 } }, 'Mermaid': { class: 'Caster', stats: { STR: 2, DEX: 4, VIT: 4, NTL: 16, WIS: 14 } }, 'Vampire': { class: 'Caster', stats: { STR: 4, DEX: 2, VIT: 16, NTL: 4, WIS: 14 } }, 'Werewolf': { class: 'Fighter', stats: { STR: 16, DEX: 12, VIT: 8, NTL: 2, WIS: 2 } }, 'Banshee': { class: 'Caster', stats: { STR: 2, DEX: 2, VIT: 6, NTL: 18, WIS: 12 } }, 'Paladin': { class: 'Caster', stats: { STR: 8, DEX: 4, VIT: 4, NTL: 12, WIS: 12 } }, 'Demon': { class: 'Caster', stats: { STR: 16, DEX: 4, VIT: 10, NTL: 16, WIS: 4 } }, 'Angel': { class: 'Caster', stats: { STR: 9, DEX: 9, VIT: 6, NTL: 16, WIS: 10 } }, },

equipmentSlotConfig: [ { name: 'Helmet', type: 'Helmet' }, { name: 'Weapon 1', type: 'Weapon' }, { name: 'Armor', type: 'Armor' }, { name: 'Weapon 2', type: 'Weapon' }, { name: 'Gauntlets', type: 'Gauntlets' }, { name: 'Leggings', type: 'Leggings' }, { name: 'Boots', type: 'Boots' }, { name: 'Necklace', type: 'Amulet' }, { name: 'Spell 1', type: 'Spellbook' }, { name: 'Ring', type: 'Ring' }, { name: 'Spell 2', type: 'Spellbook' } ],

Enchantments: {

Caster: {

"LoreStone Imbuement": { effect: "Increase Spell Class", values: [0.25, 0.5, 0.5, 1, 1.25, 2, 3.25, 4, 5] },

"MindStone Infusion": { effect: "Increase Ntl", values: [1.25, 1.875, 2.5, 3.125, 3.75, 5, 7.5, 10, 12.5] },

"Mindrite Weave": { effect: "Increase Wis", values: [1.25, 1.875, 2.5, 3.125, 3.75, 5, 7.5, 10, 12.5] },

},

Fighter: {

"WarStone Imbuement": { effect: "Increase Weapon Class", values: [0.25, 0.5, 0.75, 1.25, 1.75, 2.5, 3.25, 4, 5] },

"MightStone's Strength": { effect: "Increase Str", values: [1.25, 1.875, 2.5, 3.125, 3.75, 5, 7.5, 10, 12.5] },

"Agilite's Swiftness": { effect: "Increase Dex", values: [1.25, 1.875, 2.5, 3.125, 3.75, 5, 7.5, 10, 12.5] },

},

Misc: {

"Obsidian Ward": { effect: "Increase Armor Class", values: [0.25, 0.5, 0.75, 1.25, 1.75, 2.5, 3.25, 4, 5] },

"Spike-Core's Edge": { effect: "Increase Crit", values: [0.25, 0.625, 1.25, 1.875, 2.5, 3.125, 3.75, 4.375, 5] },

"True-Core's Aim": { effect: "Increase Hit", values: [0.75, 1.5, 2.25, 3, 3.75, 4.5, 5.25, 6.25, 7.5] },

}

},

Monsters: {

mountain: [ { id: 'm\_goblin', name: 'Mountain Goblin', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 }, { id: 'm\_troll', name: 'Rock Troll', baseHp: 54, baseAttack: 18, baseAC: 23, baseXP: 35, baseGold: 7 }, ],

forest: [ { id: 'f\_spider', name: 'Forest Spider', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 }, { id: 'f\_wolf', name: 'Dire Wolf', baseHp: 54, baseAttack: 18, baseAC: 23, baseXP: 35, baseGold: 7 }, ],

plains: [ { id: 'p\_bandit', name: 'Plains Bandit', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

wastes: [ { id: 'w\_scorpion', name: 'Giant Scorpion', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

swamp: [ { id: 's\_leech', name: 'Bloated Leech', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

jungle: [ { id: 'j\_panther', name: 'Shadow Panther', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

tundra: [ { id: 't\_yeti', name: 'Tundra Yeti', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

coastal: [ { id: 'c\_crab', name: 'Armored Crab', baseHp: 25, baseAttack: 10, baseAC: 13, baseXP: 15, baseGold: 3 } ],

},

SpecialMonsterTitles: {

// Tier 1 (Normal Rarity), Base Spawn Chance: 1 in 250

Gilded: { tier: 1, rarity: 1, effects: { goldMult: 12 } },

Echo: { tier: 1, rarity: 1, effects: { statMult: 2, goldMult: 2, dropChanceMult: 4 } },

Marauder: { tier: 1, rarity: 1, effects: { doubleHitBonus: 70 } },

// Tier 4 (Epic Rarity), 12x Rarer

Apex: { tier: 4, rarity: 12, effects: { statMult: 4, goldMult: 4, xpMult: 4, acBonus: 4, doubleHitBonus: 50, guaranteedDrop: 'Shadow' } },

Behemoth: { tier: 4, rarity: 12, effects: { statMult: 2, goldMult: 2, xpMult: 5, hpMult: 5, acBonus: 7, wcScPenalty: -2, doubleHitBonus: 20, guaranteedDrop: 'Gem' } },

// Tier 5 (Mythic Rarity)

Terminus: { tier: 5, rarity: 1000, effects: { instantDeath: true } }

}

};

// --- CHARACTER CREATION ---

const CreationManager = {

init() {

const contentHTML = `

<div class="creation-card w-full h-full flex flex-col">

<div class="flex-shrink-0">

<h1 class="text-3xl font-orbitron text-center text-gray-300 mb-4">Create Your Hero</h1>

<div class="mb-4 px-4">

<input type="text" id="creation-player-name" placeholder="Enter Character Name" class="w-full p-2 rounded text-lg bg-gray-200 placeholder-gray-500 focus:outline-none focus:ring-2 focus:ring-[var(--highlight-orange)] text-black">

</div>

<h2 class="text-xl font-orbitron text-center text-gray-300 mb-4">Choose Your Race</h2>

</div>

<div id="creation-race-grid" class="flex-grow overflow-y-auto custom-scrollbar grid grid-cols-2 md:grid-cols-4 gap-2 px-4">

${Object.keys(GameData.Races).map(raceName =>

`<div class="race-option p-3 text-center border border-transparent rounded-md cursor-pointer hover:bg-[var(--highlight-orange)]/20 font-orbitron" data-race="${raceName}">${raceName}</div>`

).join('')}

</div>

<div class="flex-shrink-0 mt-4 px-4">

<button id="finish-creation-btn" class="glass-button w-full py-3 font-bold rounded-md" disabled>Finish</button>

</div>

</div>

`;

ModalManager.show('Create Your Character', contentHTML, {

widthClass: 'w-full max-w-3xl h-full sm:h-auto sm:max-h-[90vh]',

onContentReady: (contentDiv) => {

let selectedRace = null;

const finishBtn = contentDiv.querySelector('#finish-creation-btn');

const nameInput = contentDiv.querySelector('#creation-player-name');

const checkCanFinish = () => {

const name = nameInput.value.trim();

finishBtn.disabled = !selectedRace || name.length < 3;

};

contentDiv.querySelectorAll('.race-option').forEach(option => {

option.addEventListener('click', () => {

selectedRace = option.dataset.race;

contentDiv.querySelectorAll('.race-option').forEach(el => el.classList.remove('selected', 'bg-[var(--highlight-orange)]/30'));

option.classList.add('selected', 'bg-[var(--highlight-orange)]/30');

checkCanFinish();

});

});

nameInput.addEventListener('input', checkCanFinish);

finishBtn.addEventListener('click', () => {

const playerName = nameInput.value.trim();

this.finishCreation(playerName, selectedRace);

});

}

});

},

finishCreation(playerName, raceName) {

if (!raceName || !playerName) return;

const raceData = GameData.Races[raceName];

state.player = {

name: playerName,

level: 1,

xp: 0,

xpToNextLevel: 200,

gold: 100,

bankGold: 0,

attributePoints: 40, // GDD Change

race: raceName,

class: raceData.class,

baseStats: { ...raceData.stats },

stats: {},

inventory: [],

equipment: {},

gems: [],

activeQuests: [],

questStreak: 0,

questPool: { xp: 0, gold: 0, items: [] },

defeatedBosses: []

};

GameData.equipmentSlotConfig.forEach(slot => { state.player.equipment[slot.name] = null; });

// Give player starting items

GameData.ItemFactory.baseItemTemplates.forEach(baseItem => {

const newItem = GameData.ItemFactory.createItemInstance(baseItem.id, 1, 'Dropper');

if (newItem) {

state.player.inventory.push(newItem);

}

});

// Give player starting gems

state.player.gems = Object.keys(GameData.Gems).map(gemId => ({ id: gemId, grade: 1 }));

ProfileManager.calculateAllStats();

state.player.hp = state.player.stats.maxHp;

ModalManager.hide();

GameManager.init();

}

};

// --- PROFILE & DATA MANAGER ---

const ProfileManager = {

addXp(amount) {

state.player.xp += amount;

while (state.player.xp >= state.player.xpToNextLevel) {

this.levelUp();

}

this.updateAllProfileUI();

},

addGold(amount) {

state.player.gold += amount;

this.updateAllProfileUI();

},

levelUp() {

state.player.xp -= state.player.xpToNextLevel;

state.player.level++;

// GDD FORMULA IMPLEMENTATION

state.player.xpToNextLevel = Math.floor(200 \* Math.pow(1.12, state.player.level));

state.player.attributePoints += 40; // GDD Change

showToast(`You have reached Level ${state.player.level}! You gained 40 Attribute Points.`);

this.calculateAllStats();

state.player.hp = state.player.stats.maxHp;

TeleportManager.populateZoneList();

QuestManager.assignQuests(); // Re-assign quests on level up

},

spendAttributePoint(clickedAttr) {

if (state.player.attributePoints < 40) {

showToast("You need 40 points to allocate.", true);

return;

}

const p = state.player;

const raceData = GameData.Races[p.race];

const weights = raceData.stats;

const totalWeight = Object.values(weights).reduce((sum, val) => sum + val, 0);

const pointsPerWeight = 40 / totalWeight;

let gains = {};

for (const stat in weights) {

gains[stat] = Math.round(weights[stat] \* pointsPerWeight);

}

// GDD Special Logic for VIT and Off-Stats

if (clickedAttr === 'VIT') {

const normalVitGain = gains.VIT;

const bonusVit = normalVitGain \* 0.5; // 1.5x total means 0.5 bonus

gains.VIT += Math.round(bonusVit);

const deductionAttr = p.class === 'Fighter' ? 'DEX' : 'WIS';

gains[deductionAttr] -= Math.round(bonusVit);

} else {

const mainStats = p.class === 'Fighter' ? ['STR', 'DEX'] : ['NTL', 'WIS'];

if (!mainStats.includes(clickedAttr) && clickedAttr !== 'VIT') {

const offStat = clickedAttr;

let mainStatToSwap;

if (p.class === 'Fighter') {

mainStatToSwap = (offStat === 'NTL') ? 'STR' : 'WIS';

} else { // Caster

mainStatToSwap = (offStat === 'STR') ? 'NTL' : 'DEX';

}

const offStatGain = gains[offStat];

const mainStatGain = gains[mainStatToSwap];

gains[mainStatToSwap] = offStatGain;

gains[offStat] = Math.round(mainStatGain \* 0.75); // 25% penalty

}

}

// Apply gains

for (const stat in gains) {

p.baseStats[stat] += gains[stat];

}

p.attributePoints -= 40;

this.calculateAllStats();

UIManager.flashStatUpdate(clickedAttr);

showToast("Attributes increased!", false);

},

healPlayer() {

if (state.player && state.player.stats) {

state.player.hp = state.player.stats.maxHp;

this.updateAllProfileUI();

showToast("You feel refreshed and fully healed!", false);

}

},

calculateAllStats() {

const p = state.player;

if (!p.baseStats) return;

// Step 1: Initialize stats and bonuses

const modifiedStats = { ...p.baseStats };

let baseWC = 0, baseSC = 0, baseAC = 0;

let hitChanceBonus = 0;

let classBonus = 0;

let gemBonuses = {

baseWCMult: 1, baseSCMult: 1, baseACMult: 1,

strMult: 1, dexMult: 1, vitMult: 1, ntlMult: 1, wisMult: 1,

critChanceAdd: 0, hitChanceAdd: 0,

};

// Step 2: Iterate through equipped gear to gather base stats and gem bonuses

Object.values(p.equipment).forEach(instanceId => {

if (!instanceId) return;

const item = p.inventory.find(i => i.instanceId === instanceId);

if (!item) return;

const baseItemData = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

if (!baseItemData) return;

// Add base stats from gear

if (baseItemData.type === 'Weapon') baseWC += item.classValue;

else if (baseItemData.type === 'Spellbook') baseSC += item.classValue;

else if (['Helmet', 'Armor', 'Gauntlets', 'Leggings', 'Boots', 'Amulet', 'Ring'].includes(baseItemData.type)) baseAC += item.classValue;

// Add special bonuses from gear

if (item.special) {

if (item.special.hitChanceBonus) hitChanceBonus += item.special.hitChanceBonus;

if (item.special.classBonus) classBonus += item.special.classBonus;

}

// Accumulate bonuses from socketed gems

if (item.socketedGems) {

item.socketedGems.forEach(gemInfo => {

if (!gemInfo) return;

const gemData = GameData.Gems[gemInfo.id];

if (!gemData) return;

// Determine the effective grade based on player level

let effectiveGrade = 0;

for (let i = 0; i < GameData.GemGradeUnlockLevels.length; i++) {

if (p.level >= GameData.GemGradeUnlockLevels[i]) {

effectiveGrade = i + 1;

} else {

break;

}

}

const gradeIndex = Math.min(gemInfo.grade, effectiveGrade) - 1;

if (gradeIndex < 0) return;

// Apply gem effect

const applyBonus = (effect, values) => {

if (!values) return;

const value = Array.isArray(values) ? values[gradeIndex] : (values.sc ? values.sc[gradeIndex] : values.wc[gradeIndex]);

const acValue = values.ac ? values.ac[gradeIndex] : 0;

switch (effect) {

case 'Increase Base Spell Class': gemBonuses.baseSCMult += value / 100; break;

case 'Increase Base Weapon Class': gemBonuses.baseWCMult += value / 100; break;

case 'Increase Base Armor Class': gemBonuses.baseACMult += value / 100; break;

case 'Increase Base SC & AC': gemBonuses.baseSCMult += value / 100; gemBonuses.baseACMult += acValue / 100; break;

case 'Increase Base WC & AC': gemBonuses.baseWCMult += value / 100; gemBonuses.baseACMult += acValue / 100; break;

case 'Increase Strength': gemBonuses.strMult += value / 100; break;

case 'Increase Dexterity': gemBonuses.dexMult += value / 100; break;

case 'Increase Intelligence': gemBonuses.ntlMult += value / 100; break;

case 'Increase Wisdom': gemBonuses.wisMult += value / 100; break;

case 'Increase Ntl & Wisdom': gemBonuses.ntlMult += value / 100; gemBonuses.wisMult += value / 100; break;

case 'Increase STR & DEX': gemBonuses.strMult += value / 100; gemBonuses.dexMult += value / 100; break;

case 'Increase Critical Hit Chance': gemBonuses.critChanceAdd += value; break;

case 'Increase Hit Chance': gemBonuses.hitChanceAdd += value; break;

// Debuff/Steal effects would be handled in combat logic, not here

}

};

applyBonus(gemData.effect, gemData.values);

});

}

});

// Step 3: Apply gem multipliers

baseWC \*= gemBonuses.baseWCMult;

baseSC \*= gemBonuses.baseSCMult;

baseAC \*= gemBonuses.baseACMult;

modifiedStats.STR \*= gemBonuses.strMult;

modifiedStats.DEX \*= gemBonuses.dexMult;

modifiedStats.VIT \*= gemBonuses.vitMult;

modifiedStats.NTL \*= gemBonuses.ntlMult;

modifiedStats.WIS \*= gemBonuses.wisMult;

// Step 4: Calculate final stats using GDD formulas

p.stats = p.stats || {};

p.stats.finalWC = baseWC \* (1 + (modifiedStats.STR \* 0.0055)) \* (1 + classBonus);

p.stats.finalSC = baseSC \* (1 + (modifiedStats.NTL \* 0.0055)) \* (1 + classBonus);

p.stats.finalAC = baseAC \* (1 + (modifiedStats.VIT \* 0.0075));

p.stats.maxHp = 100 + (modifiedStats.VIT \* 10);

const hitAttribute = p.class === 'Fighter' ? modifiedStats.DEX : modifiedStats.WIS;

p.stats.hitChance = (90 + (hitAttribute \* 0.05) + gemBonuses.hitChanceAdd) \* (1 + hitChanceBonus);

p.stats.critChance = 5 + (hitAttribute \* 0.01) + gemBonuses.critChanceAdd;

if (p.hp > p.stats.maxHp) p.hp = p.stats.maxHp;

this.updateAllProfileUI();

},

updateAllProfileUI() {

if (!state.player || !state.player.name || !state.player.stats) return;

UIManager.updatePlayerStatusUI();

if (GameManager.isInitialized) UIManager.updateCombatStatsUI();

if (GameManager.isInitialized) StatsManager.renderStats();

}

};

const StatsManager = {

isInitialized: false,

statMetadata: {

STR: { name: 'Strength', icon: '⚔️', description: 'Increases physical damage from Fighter class weapons and contributes to carrying capacity.' },

DEX: { name: 'Dexterity', icon: '🏹', description: 'Improves accuracy, critical hit chance, and effectiveness of finesse-based weapons.' },

VIT: { name: 'Vitality', icon: '❤️', description: 'Increases maximum Health Points and improves resistance to physical damage.' },

NTL: { name: 'Intelligence', icon: '🧠', description: 'Boosts magical damage from Caster class spellbooks and increases maximum Mana.' },

WIS: { name: 'Wisdom', icon: '👁️', description: 'Enhances magical accuracy, critical spell chance, and resistance to magical effects.' },

finalWC: { name: 'Weapon Class', icon: '💥', description: 'Your total effectiveness with physical weapons, calculated from Strength and equipped items.' },

finalSC: { name: 'Spell Class', icon: '🔮', description: 'Your total effectiveness with magic, calculated from Intelligence and equipped spellbooks.' },

finalAC: { name: 'Armor Class', icon: '🛡️', description: 'Your total damage reduction, calculated from Vitality and equipped armor.' },

maxHp: { name: 'Health Points', icon: '❤️', description: 'Your life force. If it reaches zero, you are defeated.' },

hitChance: { name: 'Hit Chance', icon: '🎯', description: 'The probability of successfully landing an attack on an enemy.' },

critChance: { name: 'Crit Chance', icon: '✨', description: 'The probability of an attack dealing bonus critical damage.' },

},

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.render();

this.addEventListeners();

},

render() {

ui.tabContentStats.innerHTML = `

<div id="stats-container" class="space-y-3">

<div class="stat-accordion-item open">

<button class="stat-accordion-header">

<h3 class="font-orbitron">Secondary Attributes</h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content">

<div id="secondary-stats-list"></div>

<div class="stat-line mt-2">

<span class="stat-icon">💎</span>

<span class="stat-name">Unspent Points</span>

<span id="unspent-points-value" class="stat-value text-[var(--highlight-orange)]">0</span>

</div>

</div>

</div>

<div class="stat-accordion-item">

<button class="stat-accordion-header">

<h3 class="font-orbitron">Primary Combat Stats</h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content">

<div id="primary-stats-list"></div>

</div>

</div>

<div class="stat-accordion-item">

<button class="stat-accordion-header">

<h3 class="font-orbitron">Derived Stats</h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content">

<div id="derived-stats-list"></div>

</div>

</div>

</div>`;

this.renderStats();

},

addEventListeners() {

ui.tabContentStats.addEventListener('click', (e) => {

const header = e.target.closest('.stat-accordion-header');

const attrBtn = e.target.closest('.attr-btn');

const infoBtn = e.target.closest('.info-btn');

if (header) {

header.parentElement.classList.toggle('open');

} else if (attrBtn) {

ProfileManager.spendAttributePoint(attrBtn.dataset.attr);

} else if (infoBtn) {

this.showStatInfo(infoBtn.dataset.title, infoBtn.dataset.description);

}

});

document.getElementById('stat-info-backdrop').addEventListener('click', () => this.hideStatInfo());

},

renderStats() {

const p = state.player;

if (!p || !p.stats || !p.baseStats) return;

const canUpgrade = p.attributePoints >= 40;

const secondaryList = ui.tabContentStats.querySelector('#secondary-stats-list');

const primaryList = ui.tabContentStats.querySelector('#primary-stats-list');

const derivedList = ui.tabContentStats.querySelector('#derived-stats-list');

const unspentPointsValue = ui.tabContentStats.querySelector('#unspent-points-value');

if (!secondaryList || !primaryList || !derivedList || !unspentPointsValue) return;

unspentPointsValue.textContent = p.attributePoints || 0;

const createStatLine = (attrKey, value, isUpgradable = false) => {

const meta = this.statMetadata[attrKey] || { name: attrKey, icon: '❓', description: 'No info available.' };

const upgradeButton = isUpgradable ? `<button class="attr-btn" data-attr="${attrKey}" ${!canUpgrade ? 'disabled' : ''}>+</button>` : '';

const infoButton = `<button class="info-btn" data-title="${meta.name}" data-description="${meta.description}">ⓘ</button>`;

return `

<div class="stat-line">

<span class="stat-icon">${meta.icon}</span>

<span class="stat-name">${meta.name}</span>

<span class="stat-value" data-stat-value="${attrKey}">${value}</span>

${upgradeButton}

${infoButton}

</div>`;

};

const createHpLine = () => {

const meta = this.statMetadata.maxHp;

const hpPercent = (p.hp / p.stats.maxHp) \* 100;

return `

<div class="stat-line">

<span class="stat-icon">${meta.icon}</span>

<span class="stat-name">${meta.name}</span>

<div class="flex-grow flex items-center gap-2">

<div class="progress-bar-track h-3 flex-grow"><div class="progress-bar-fill h-full" style="width: ${hpPercent}%; background-color: var(--hp-color);"></div></div>

<span class="stat-value">${Math.ceil(p.hp)} / ${Math.ceil(p.stats.maxHp)}</span>

</div>

<button class="info-btn" data-title="${meta.name}" data-description="${meta.description}">ⓘ</button>

</div>`;

};

secondaryList.innerHTML = `

${createStatLine('STR', Math.round(p.baseStats.STR), true)}

${createStatLine('DEX', Math.round(p.baseStats.DEX), true)}

${createStatLine('VIT', Math.round(p.baseStats.VIT), true)}

${createStatLine('NTL', Math.round(p.baseStats.NTL), true)}

${createStatLine('WIS', Math.round(p.baseStats.WIS), true)}

`;

primaryList.innerHTML = `

${createStatLine('finalWC', p.stats.finalWC.toFixed(2))}

${createStatLine('finalSC', p.stats.finalSC.toFixed(2))}

${createStatLine('finalAC', p.stats.finalAC.toFixed(2))}

`;

derivedList.innerHTML = `

${createHpLine()}

${createStatLine('hitChance', `${p.stats.hitChance.toFixed(2)}%`)}

${createStatLine('critChance', `${p.stats.critChance.toFixed(2)}%`)}

`;

},

showStatInfo(title, description) {

const modal = document.getElementById('stat-info-modal');

modal.querySelector('#stat-info-title').textContent = title;

modal.querySelector('#stat-info-description').textContent = description;

modal.style.display = 'flex';

},

hideStatInfo() {

document.getElementById('stat-info-modal').style.display = 'none';

}

};

const EquipmentManager = {

isInitialized: false,

filterState: {

category: 'All',

subType: 'All',

tier: 'All',

quality: 'All',

sortBy: 'tier',

order: 'desc'

},

inventoryBags: {

'Weapon Chest': ['Weapon'],

'Bag of Gear': ['Helmet', 'Armor', 'Leggings', 'Boots', 'Gauntlets'],

'Jewelry Box': ['Amulet', 'Ring'],

'Spell Satchel': ['Spellbook'],

},

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.renderEquipmentTab();

this.renderInventoryTab();

this.addEventListeners();

this.updateAllViews();

this.populateFilterOptions();

},

renderEquipmentTab() {

ui.tabContentEquipment.innerHTML = `

<div class="flex gap-2 mb-2">

<button class="glass-button flex-1 py-1 text-sm rounded-md active" data-view="equipment">Equipment</button>

<button class="glass-button flex-1 py-1 text-sm rounded-md" data-view="socket">Socket</button>

</div>

<div id="equipment-view-content"></div>

`;

},

renderInventoryTab() {

let bagsHTML = '';

for (const bagName in this.inventoryBags) {

bagsHTML += `

<div class="stat-accordion-item" data-bag-container="${bagName}">

<button class="stat-accordion-header">

<h3>${bagName} <span id="inventory-${bagName.replace(/\s+/g, '-')}-count" class="text-xs text-gray-500 font-sans"></span></h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content !p-2">

<div class="inventory-grid" data-bag-name="${bagName}"></div>

</div>

</div>`;

}

ui.tabContentInventory.innerHTML = `

<div id="inventory-sort-container" class="mb-2">

<div class="stat-accordion-item open">

<button class="stat-accordion-header">

<h3><svg class="w-5 h-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M3 4h13M3 8h9M3 12h9m-9 4h6"></path></svg>Sort & Filter</h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content !p-2">

<div class="grid grid-cols-2 md:grid-cols-4 gap-2">

<div>

<label class="text-xs text-gray-400">Category</label>

<select id="inventory-filter-category-select" class="editor-input !w-full !text-xs"></select>

</div>

<div>

<label class="text-xs text-gray-400">Sub-Type</label>

<select id="inventory-filter-subtype-select" class="editor-input !w-full !text-xs"></select>

</div>

<div>

<label class="text-xs text-gray-400">Tier</label>

<select id="inventory-filter-tier-select" class="editor-input !w-full !text-xs"></select>

</div>

<div>

<label class="text-xs text-gray-400">Quality</label>

<select id="inventory-filter-quality-select" class="editor-input !w-full !text-xs"></select>

</div>

</div>

<div class="grid grid-cols-2 gap-2 mt-2 border-t border-gray-700 pt-2">

<div>

<label class="text-xs text-gray-400">Sort By</label>

<select id="inventory-sort-by-select" class="editor-input !w-full !text-xs">

<option value="tier">Tier</option>

<option value="name">Name</option>

<option value="type">Type</option>

</select>

</div>

<div>

<label class="text-xs text-gray-400">Order</label>

<select id="inventory-sort-order-select" class="editor-input !w-full !text-xs">

<option value="desc">Descending</option>

<option value="asc">Ascending</option>

</select>

</div>

</div>

</div>

</div>

</div>

${bagsHTML}

<div class="stat-accordion-item open">

<button class="stat-accordion-header">

<h3>Gem Pouch <span id="inventory-gem-pouch-count" class="text-xs text-gray-500 font-sans"></span></h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content !p-2">

<div class="gem-pouch-grid"></div>

</div>

</div>`;

},

addEventListeners() {

ui.tabContentInventory.addEventListener('click', e => {

const slot = e.target.closest('.inventory-slot');

const gemItem = e.target.closest('.gem-item');

const header = e.target.closest('.stat-accordion-header');

if (slot && slot.dataset.instanceId) this.showItemActionModal(slot.dataset.instanceId, null);

else if (gemItem && gemItem.dataset.gemId) this.showItemActionModal(null, gemItem.dataset.gemId);

else if (header) header.parentElement.classList.toggle('open');

});

ui.tabContentInventory.addEventListener('change', e => {

const targetId = e.target.id;

if (targetId.startsWith('inventory-filter-') || targetId.startsWith('inventory-sort-')) {

this.filterState.category = document.getElementById('inventory-filter-category-select').value;

this.filterState.subType = document.getElementById('inventory-filter-subtype-select').value;

this.filterState.tier = document.getElementById('inventory-filter-tier-select').value;

this.filterState.quality = document.getElementById('inventory-filter-quality-select').value;

this.filterState.sortBy = document.getElementById('inventory-sort-by-select').value;

this.filterState.order = document.getElementById('inventory-sort-order-select').value;

if (targetId === 'inventory-filter-category-select') {

this.populateSubTypeFilter();

this.filterState.subType = 'All';

document.getElementById('inventory-filter-subtype-select').value = 'All';

}

this.renderInventoryBags();

}

});

ui.tabContentEquipment.addEventListener('click', e => {

if (e.target.closest('[data-view]')) this.switchEquipmentView(e.target.closest('[data-view]').dataset.view);

else if (e.target.closest('.gem-socket')) this.handleSocketClick(e.target.closest('.gem-socket'));

else if (e.target.closest('.equipment-slot-content')?.dataset.instanceId) this.showItemActionModal(e.target.closest('.equipment-slot-content').dataset.instanceId, null);

});

ui.itemActionModalBackdrop.addEventListener('click', () => this.hideItemActionModal());

},

populateFilterOptions() {

const categorySelect = document.getElementById('inventory-filter-category-select');

let categories = ['All', ...Object.keys(this.inventoryBags)];

categorySelect.innerHTML = categories.map(c => `<option value="${c}">${c}</option>`).join('');

const tierSelect = document.getElementById('inventory-filter-tier-select');

let tierOptions = '<option value="All">All Tiers</option>';

for(let i = 1; i <= 20; i++) {

tierOptions += `<option value="${i}">Tier ${i}</option>`;

}

tierSelect.innerHTML = tierOptions;

const qualitySelect = document.getElementById('inventory-filter-quality-select');

const qualities = ['All', 'Dropper', 'Shadow', 'Echo'];

qualitySelect.innerHTML = qualities.map(q => `<option value="${q}">${q}</option>`).join('');

this.populateSubTypeFilter();

},

populateSubTypeFilter() {

const category = document.getElementById('inventory-filter-category-select').value;

const subTypeSelect = document.getElementById('inventory-filter-subtype-select');

let subTypes = new Set();

const itemsToScan = GameData.ItemFactory.baseItemTemplates.filter(item => {

if (category === 'All') return true;

const typesInBag = this.inventoryBags[category];

return typesInBag && typesInBag.includes(item.type);

});

itemsToScan.forEach(item => {

if ((item.type === 'Weapon' || item.type === 'Spellbook') && item.subType) {

subTypes.add(item.subType);

}

else if (item.type !== 'Weapon' && item.type !== 'Spellbook') {

subTypes.add(item.type);

}

});

if (category === 'All') {

GameData.ItemFactory.baseItemTemplates.forEach(item => {

subTypes.add(item.type);

});

}

let specificTypes = Array.from(subTypes);

specificTypes.sort();

const finalOptions = ['All', ...specificTypes];

subTypeSelect.innerHTML = finalOptions.map(s => `<option value="${s}">${s}</option>`).join('');

},

renderInventoryBags() {

const equippedIds = Object.values(state.player.equipment).filter(Boolean);

let unequippedItems = state.player.inventory.filter(item => !equippedIds.includes(item.instanceId));

const { category, subType, tier, quality } = this.filterState;

const filteredItems = unequippedItems.filter(item => {

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

if (!base) return false;

if (category !== 'All' && !this.inventoryBags[category]?.includes(base.type)) {

return false;

}

if (subType !== 'All') {

if (base.subType !== subType && base.type !== subType) {

return false;

}

}

if (tier !== 'All' && item.tier.toString() !== tier) {

return false;

}

if (quality !== 'All' && item.type !== quality) {

return false;

}

return true;

});

filteredItems.sort((a, b) => {

const baseA = GameData.ItemFactory.baseItemTemplates.find(item => item.id === a.baseItemId);

const baseB = GameData.ItemFactory.baseItemTemplates.find(item => item.id === b.baseItemId);

let compareA, compareB;

switch (this.filterState.sortBy) {

case 'name': compareA = baseA.name; compareB = baseB.name; break;

case 'type': compareA = baseA.type; compareB = baseB.type; break;

default: compareA = a.tier; compareB = b.tier; break;

}

if (compareA < compareB) return this.filterState.order === 'asc' ? -1 : 1;

if (compareA > compareB) return this.filterState.order === 'asc' ? 1 : -1;

return 0;

});

document.querySelectorAll('#tab-content-inventory .inventory-grid').forEach(grid => grid.innerHTML = '');

filteredItems.forEach(item => {

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

for (const bagName in this.inventoryBags) {

if (this.inventoryBags[bagName].includes(base.type)) {

const grid = document.querySelector(`#tab-content-inventory .inventory-grid[data-bag-name="${bagName}"]`);

if (grid) {

let itemStyle = item.type === 'Shadow' ? 'color: #a0a0a0;' : 'color: var(--text-primary);';

let gemDotsHTML = '';

if (item.socketedGems && item.socketedGems.filter(g => g).length > 0) {

gemDotsHTML = `<div class="gem-dot-container">${item.socketedGems.filter(g => g).map(() => `<div class="gem-dot"></div>`).join('')}</div>`;

}

const itemHTML = `

<div class="inventory-slot" data-instance-id="${item.instanceId}">

${gemDotsHTML}

<img src="${base.imageUrl}" onerror="this.onerror=null;this.src='https://placehold.co/60x60/1f2937/ffffff?text=ERR';">

<span class="item-label" style="${itemStyle}">T${item.tier}</span>

</div>`;

grid.innerHTML += itemHTML;

}

break;

}

}

});

const isAnyFilterActive = category !== 'All' || subType !== 'All' || tier !== 'All' || quality !== 'All';

document.querySelectorAll('#tab-content-inventory .stat-accordion-item[data-bag-container]').forEach(container => {

const bagName = container.dataset.bagContainer;

const grid = container.querySelector(`.inventory-grid[data-bag-name="${bagName}"]`);

if (isAnyFilterActive) {

if (grid && grid.children.length === 0) {

container.style.display = 'none';

} else {

container.style.display = 'block';

}

} else {

container.style.display = 'block';

}

});

this.updateCounts();

},

updateAllViews() {

if (this.isInitialized) {

this.renderEquipmentView();

this.renderInventoryBags();

this.renderGemPouch();

this.updateCounts();

}

},

updateCounts() {

const equippedIds = Object.values(state.player.equipment).filter(Boolean);

const unequippedItems = state.player.inventory.filter(item => !equippedIds.includes(item.instanceId));

for (const bagName in this.inventoryBags) {

const itemTypesInBag = this.inventoryBags[bagName];

const count = unequippedItems.filter(item => {

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

return baseItem && itemTypesInBag.includes(baseItem.type);

}).length;

const countEl = document.getElementById(`inventory-${bagName.replace(/\s+/g, '-')}-count`);

if (countEl) countEl.textContent = `(${count})`;

}

const gemPouchCountEl = document.getElementById('inventory-gem-pouch-count');

if (gemPouchCountEl) {

const gemCount = state.player.gems.length;

gemPouchCountEl.textContent = `(${gemCount}/${UIManager.MAX\_GEMS})`;

}

},

switchEquipmentView(view) {

ui.tabContentEquipment.querySelectorAll('[data-view]').forEach(btn => btn.classList.remove('active'));

ui.tabContentEquipment.querySelector(`[data-view="${view}"]`).classList.add('active');

if (view === 'equipment') {

this.renderEquipmentView();

} else {

this.renderSocketView();

}

},

renderEquipmentView() {

const content = ui.tabContentEquipment.querySelector('#equipment-view-content');

if (!content || !state.player.equipment) return;

const slotsHTML = GameData.equipmentSlotConfig.map(slot => {

const instanceId = state.player.equipment[slot.name];

const item = state.player.inventory.find(i => i.instanceId === instanceId);

let contentHTML = `<span class="text-xs text-gray-500">Empty</span>`;

if (item) {

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

let color = item.type === 'Shadow' ? '#a0a0a0' : 'var(--text-primary)';

let gemDotsHTML = '';

if (item.socketedGems && item.socketedGems.filter(g => g).length > 0) {

gemDotsHTML = `<div class="gem-dot-container">${item.socketedGems.filter(g => g).map(() => `<div class="gem-dot"></div>`).join('')}</div>`;

}

// --- MODIFICATION START ---

let gemListHTML = '';

if (item.socketedGems && item.socketedGems.filter(g => g).length > 0) {

gemListHTML = `<div class="equipment-gem-list">`;

item.socketedGems.forEach(gemInfo => {

if (gemInfo) {

const gemData = GameData.Gems[gemInfo.id];

gemListHTML += `<div>${gemData.abbreviation}${gemInfo.grade}</div>`;

}

});

gemListHTML += `</div>`;

}

// --- MODIFICATION END ---

contentHTML = `

${gemDotsHTML}

${gemListHTML}

<img src="${base.imageUrl}" class="h-12 w-12 object-contain" onerror="this.onerror=null;this.src='https://placehold.co/48x48/1f2937/ffffff?text=ERR';">

<span class="item-label" style="color: ${color};">T${item.tier}</span>`;

}

return `

<div class="equipment-slot-wrapper">

<div class="equipment-slot-title font-orbitron"><span>${slot.name}</span></div>

<div class="equipment-slot-content" data-slot-name="${slot.name}" data-instance-id="${instanceId || ''}">${contentHTML}</div>

</div>`;

}).join('');

content.innerHTML = `<div class="equipment-grid">${slotsHTML}</div>`;

},

renderSocketView() {

const content = ui.tabContentEquipment.querySelector('#equipment-view-content');

const item = state.player.inventory.find(i => i.instanceId === state.ui.selectedInventoryId);

if (!item) {

content.innerHTML = `<div class="text-center p-8 font-orbitron">Select an item from your inventory to socket gems.</div>`;

return;

}

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

if (!base.sockets || base.sockets === 0) {

content.innerHTML = `<div class="text-center p-8 font-orbitron">${base.name} has no sockets.</div>`;

return;

}

const socketsHTML = Array(base.sockets).fill(0).map((\_, i) => {

const gemInfo = item.socketedGems[i];

const gemData = gemInfo ? GameData.Gems[gemInfo.id] : null;

return ` <div class="gem-socket ${gemInfo ? 'has-gem' : ''}" data-socket-index="${i}" data-gem-id="${gemInfo ? gemInfo.id : ''}"> ${gemData ? `<img src="${gemData.imageUrl}" class="w-10 h-10" onerror="this.onerror=null;this.src='https://placehold.co/40x40/1f2937/ffffff?text=ERR';"><span class="item-label">${gemData.abbreviation}${gemInfo.grade}</span>` : ''} </div>`;

}).join('');

content.innerHTML = ` <div class="text-center mb-4"> <h3 class="font-orbitron">${base.name}</h3> <div class="flex justify-center items-center gap-4 mt-2">${socketsHTML}</div> </div> `;

},

renderGemPouch() {

const grid = ui.tabContentInventory.querySelector('.gem-pouch-grid');

if (!grid || !state.player.gems) return;

grid.innerHTML = state.player.gems.map(gemInfo => {

const gem = GameData.Gems[gemInfo.id];

return ` <div class="gem-item ${gemInfo.id === state.ui.selectedGemId ? 'selected' : ''}" data-gem-id="${gemInfo.id}"> <img src="${gem.imageUrl}" class="w-10 h-10" onerror="this.onerror=null;this.src='https://placehold.co/40x40/1f2937/ffffff?text=ERR';"> <span class="item-label">${gem.abbreviation}${gemInfo.grade}</span> </div>`;

}).join('');

},

handleGemPouchSelect(gemId) {

state.ui.selectedGemId = state.ui.selectedGemId === gemId ? null : gemId;

this.renderGemPouch();

},

handleSocketClick(socketElement) {

const item = state.player.inventory.find(i => i.instanceId === state.ui.selectedInventoryId);

if (!item) return;

const socketIndex = parseInt(socketElement.dataset.socketIndex);

const existingGemInfo = item.socketedGems[socketIndex];

if (existingGemInfo) {

item.socketedGems[socketIndex] = null;

state.player.gems.push(existingGemInfo);

} else if (state.ui.selectedGemId) {

const gemToSocket = state.player.gems.find(g => g.id === state.ui.selectedGemId);

item.socketedGems[socketIndex] = gemToSocket;

state.player.gems = state.player.gems.filter(g => g.id !== state.ui.selectedGemId);

state.ui.selectedGemId = null;

}

item.socketedGems = item.socketedGems.filter(g => g);

this.renderSocketView();

this.renderGemPouch();

ProfileManager.calculateAllStats();

},

showItemActionModal(instanceId, gemId) {

const modalBody = ui.itemActionModalBody;

let contentHTML = '';

let actionButtonHTML = '';

let actionHandler = null;

if (instanceId) {

const item = state.player.inventory.find(i => i.instanceId === instanceId);

if (!item) return;

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

const isEquipped = Object.values(state.player.equipment).includes(instanceId);

const actionText = isEquipped ? 'Unequip' : 'Equip';

let gemListHTML = '';

if (item.socketedGems && item.socketedGems.filter(g => g).length > 0) {

gemListHTML = `<div class="item-gem-list">`;

item.socketedGems.forEach(gemInfo => {

if (gemInfo) {

const gemData = GameData.Gems[gemInfo.id];

const effectText = this.getGemEffectValueText(gemInfo, gemData);

gemListHTML += `

<div class="item-gem-entry">

<span class="item-gem-name">${gemData.abbreviation}${gemInfo.grade}</span>

<span class="item-gem-effect">${effectText}</span>

</div>`;

}

});

gemListHTML += `</div>`;

}

contentHTML = `

<div class="item-name">${baseItem.name}</div>

<div class="item-type">Tier ${item.tier} ${baseItem.type} (${baseItem.subType || item.type})</div>

<div class="item-stat">

<span class="item-stat-label">Class Value: </span>

<span class="item-stat-value">${item.classValue.toFixed(2)}</span>

</div>

${gemListHTML}

`;

actionButtonHTML = `<button id="item-action-button" class="glass-button w-full py-2 rounded-md">${actionText}</button>`;

actionHandler = () => {

if (isEquipped) {

const slotName = Object.keys(state.player.equipment).find(key => state.player.equipment[key] === instanceId);

this.unequipItem(slotName);

} else {

this.equipItem(instanceId);

}

};

} else if (gemId) {

const gemInfo = state.player.gems.find(g => g.id === gemId);

if (!gemInfo) return;

const gemData = GameData.Gems[gemId];

const effectValueText = this.getGemEffectValueText(gemInfo, gemData);

contentHTML = `

<div class="item-name">${gemData.name}</div>

<div class="item-type">Grade ${gemInfo.grade} Gem</div>

<div class="item-stat">

<span class="item-stat-label">${gemData.effect}: </span>

<span class="item-stat-value text-[var(--highlight-orange)]">${effectValueText}</span>

</div>

`;

actionButtonHTML = '';

actionHandler = null;

}

if (!contentHTML) return;

modalBody.innerHTML = contentHTML + actionButtonHTML;

ui.itemActionModalContent.style.display = 'block';

ui.itemActionModalBackdrop.style.display = 'block';

const actionButton = document.getElementById('item-action-button');

if (actionButton && actionHandler) {

actionButton.addEventListener('click', actionHandler, { once: true });

}

},

getGemEffectValueText(gemInfo, gemData) {

let effectiveGrade = 0;

for (let i = 0; i < GameData.GemGradeUnlockLevels.length; i++) {

if (state.player.level >= GameData.GemGradeUnlockLevels[i]) {

effectiveGrade = i + 1;

} else {

break;

}

}

const gradeIndex = Math.min(gemInfo.grade, effectiveGrade) - 1;

if (gradeIndex < 0 || !gemData.values) return 'N/A';

if (Array.isArray(gemData.values)) {

return `+${gemData.values[gradeIndex]}%`;

} else if (typeof gemData.values === 'object') {

const wcText = gemData.values.wc ? `WC: +${gemData.values.wc[gradeIndex]}%` : '';

const scText = gemData.values.sc ? `SC: +${gemData.values.sc[gradeIndex]}%` : '';

const acText = gemData.values.ac ? `AC: +${gemData.values.ac[gradeIndex]}%` : '';

return [wcText, scText, acText].filter(Boolean).join(', ');

}

return 'N/A';

},

hideItemActionModal() {

ui.itemActionModalContent.style.display = 'none';

ui.itemActionModalBackdrop.style.display = 'none';

ui.itemActionModalBody.innerHTML = '';

},

equipItem(instanceId) {

const itemInstance = state.player.inventory.find(i => i.instanceId === instanceId);

if (!itemInstance) return;

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === itemInstance.baseItemId);

if (!baseItem) return;

const slotsToOccupy = GameData.equipmentSlotConfig.filter(slot => slot.type === baseItem.type);

let targetSlot = null;

for(const slot of slotsToOccupy) {

if (!state.player.equipment[slot.name]) {

targetSlot = slot.name;

break;

}

}

if (!targetSlot && slotsToOccupy.length > 0) {

targetSlot = slotsToOccupy[0].name;

this.unequipItem(targetSlot);

}

if (!targetSlot) {

showToast(`No available slot for ${baseItem.type}.`, true);

return;

}

state.player.equipment[targetSlot] = itemInstance.instanceId;

this.hideItemActionModal();

this.updateAllViews();

ProfileManager.calculateAllStats();

showToast(`${baseItem.name} equipped.`, false);

},

unequipItem(slotName) {

if (!slotName || !state.player.equipment[slotName]) return;

const itemInstanceId = state.player.equipment[slotName];

const item = state.player.inventory.find(i => i.instanceId === itemInstanceId);

const baseItem = item ? GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId) : null;

state.player.equipment[slotName] = null;

this.hideItemActionModal();

this.updateAllViews();

ProfileManager.calculateAllStats();

if (baseItem) {

showToast(`${baseItem.name} unequipped.`, false);

}

}

};

const CombatManager = {

isInitialized: false,

currentMonster: null,

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.render();

this.addEventListeners();

},

render() {

ui.tabContentCombat.innerHTML = `

<div class="space-y-4 flex flex-col items-center h-full">

<div id="combat-info-panel" class="w-full mb-3 p-2 rounded-lg bg-black/20 border border-[var(--border-color-main)]">

<div id="combat-stats-container">

<div class="location-info">

<span class="stats-label label-location">Location:</span>

<span id="location-value" class="stats-value">Starting Zone</span>

</div>

<div class="stats-grid">

<div class="stats-col">

<div><span class="stats-label label-health">Health:</span><span id="health-value" class="stats-value">0 / 0</span></div>

<div><span class="stats-label label-exp">Experience:</span><span id="exp-value" class="stats-value">0</span></div>

<div><span class="stats-label label-next-level">Next Lvl:</span><span id="next-lvl-value" class="stats-value">0</span></div>

<div><span class="stats-label label-drop">Last Drop:</span><span id="last-drop-value" class="stats-value">None</span></div>

</div>

<div class="stats-col text-right">

<div><span class="stats-label label-level">Level:</span><span id="level-value" class="stats-value">0</span></div>

<div><span class="stats-label label-gold">Gold:</span><span id="gold-value" class="stats-value">0</span></div>

<div><span class="stats-label label-drop">Gem Drop:</span><span id="gem-drop-value" class="stats-value">None</span></div>

<div><span class="stats-label label-drop">Inv/Gems:</span><span id="inv-value" class="stats-value">0/0 | 0/0</span></div>

</div>

</div>

</div>

<div id="combat-log" class="text-gray-400 truncate mt-2 text-sm px-2"><strong>Log:</strong> Welcome to Geminus!</div>

</div>

<div class="flex-grow w-full flex flex-col space-y-3">

<select id="monsterSelect" class="w-full"></select>

<div id="monster-info" class="text-center invisible min-h-[60px]">

<div id="enemy-defeated-msg">ENEMY DEFEATED</div>

<h3 id="monster-name" class="font-orbitron text-lg"></h3>

<div class="progress-bar-track h-4 mt-2 w-full max-w-xs mx-auto">

<div id="monster-hp-bar" class="progress-bar-fill h-full" style="width: 100%; background-color: var(--hp-color);"></div>

</div>

<div id="monster-hp-text" class="text-sm mt-1"></div>

</div>

<div class="grid grid-cols-1 sm:grid-cols-2 gap-2 mt-auto">

<button class="glass-button py-2 rounded-md" id="engageBtn" disabled>🎯 ENGAGE</button>

<button class="glass-button py-2 rounded-md" id="attackBtn" disabled>⚔️ ATTACK</button>

<button class="glass-button py-2 rounded-md" id="castBtn" disabled style="display: none;">🔮 CAST</button>

</div>

</div>

</div>`;

},

addEventListeners() {

const combatTab = ui.tabContentCombat;

combatTab.querySelector('#monsterSelect').addEventListener('change', (e) => this.selectMonster(e.target.value));

combatTab.querySelector('#engageBtn').addEventListener('click', () => this.engage());

combatTab.querySelector('#attackBtn').addEventListener('click', () => this.attack());

combatTab.querySelector('#castBtn').addEventListener('click', () => this.cast());

},

resetCombatSelection() {

this.currentMonster = null;

state.game.combatActive = false;

ui.tabContentCombat.querySelector('#monster-info').classList.add('invisible');

this.updateButtons();

},

clearMonsterList() {

const monsterSelect = ui.tabContentCombat.querySelector('#monsterSelect');

monsterSelect.innerHTML = `<option value="">No monsters nearby...</option>`;

this.resetCombatSelection();

},

populateMonsterList(zoneId) {

const zoneData = AllZones[zoneId];

if (!zoneData) return this.clearMonsterList();

const biome = zoneData.biome;

const monsterTemplates = GameData.Monsters[biome];

const monsterSelect = ui.tabContentCombat.querySelector('#monsterSelect');

if (!monsterTemplates || monsterTemplates.length === 0) {

monsterSelect.innerHTML = `<option value="">No monsters in this biome...</option>`;

this.resetCombatSelection();

return;

}

let optionsHTML = '<option value="">Select a monster...</option>';

monsterTemplates.forEach(monster => {

optionsHTML += `<option value="${monster.id}">${monster.name} (T${zoneData.gearTier})</option>`;

});

monsterSelect.innerHTML = optionsHTML;

this.resetCombatSelection();

},

selectMonster(monsterId) {

if (!monsterId) {

this.resetCombatSelection();

return;

}

const zoneTier = state.game.currentZoneTier;

const zoneData = AllZones[zoneTier];

const gearTier = zoneData.gearTier;

const biome = zoneData.biome;

const monsterTemplate = GameData.Monsters[biome].find(m => m.id === monsterId);

if (!monsterTemplate) return;

// GDD Monster Scaling

let finalHp = monsterTemplate.baseHp \* Math.pow(1.20, gearTier - 1);

let finalAc = monsterTemplate.baseAC \* Math.pow(1.20, gearTier - 1);

let finalAttack = monsterTemplate.baseAttack \* Math.pow(1.22, gearTier - 1);

let finalXp = monsterTemplate.baseXP \* Math.pow(1.27, gearTier - 1);

let finalGold = monsterTemplate.baseGold \* Math.pow(1.27, gearTier - 1);

let monsterName = `${monsterTemplate.name} (T${gearTier})`;

let guaranteedDrop = null;

let isBoss = monsterId.endsWith('\*');

// GDD Special Monster Title Logic

if (!isBoss && Math.random() < 1 / 250) {

const titles = Object.entries(GameData.SpecialMonsterTitles);

const randomTitleData = titles[Math.floor(Math.random() \* titles.length)];

const [titleName, titleInfo] = randomTitleData;

monsterName += ` [${titleName}]`;

if (titleInfo.effects.statMult) {

finalHp \*= titleInfo.effects.statMult;

finalAc \*= titleInfo.effects.statMult;

finalAttack \*= titleInfo.effects.statMult;

}

if (titleInfo.effects.hpMult) finalHp \*= titleInfo.effects.hpMult;

if (titleInfo.effects.goldMult) finalGold \*= titleInfo.effects.goldMult;

if (titleInfo.effects.xpMult) finalXp \*= titleInfo.effects.xpMult;

if (titleInfo.effects.guaranteedDrop) guaranteedDrop = titleInfo.effects.guaranteedDrop;

showToast(`A special monster appears: ${monsterName}!`, false);

}

this.currentMonster = {

...monsterTemplate,

id: monsterId,

name: monsterName,

hp: finalHp,

maxHp: finalHp,

AC: finalAc,

attack: finalAttack,

xpValue: finalXp,

goldValue: finalGold,

guaranteedDrop: guaranteedDrop,

isBoss: isBoss,

zoneId: zoneTier

};

const monsterInfo = ui.tabContentCombat.querySelector('#monster-info');

monsterInfo.querySelector('#monster-name').textContent = this.currentMonster.name;

monsterInfo.classList.remove('invisible');

this.updateEnemyUI();

this.engage(true);

},

engage(isInitialSetup = false) {

if (!this.currentMonster) return;

const defeatedMsg = document.getElementById('enemy-defeated-msg');

const monsterName = document.getElementById('monster-name');

const hpBar = document.getElementById('monster-hp-bar').parentElement;

const hpText = document.getElementById('monster-hp-text');

defeatedMsg.style.display = 'none';

monsterName.style.display = 'block';

hpBar.style.display = 'block';

hpText.style.display = 'block';

if (isInitialSetup) {

this.updateButtons();

return;

}

state.game.combatActive = true;

this.currentMonster.hp = this.currentMonster.maxHp;

this.updateEnemyUI();

this.updateButtons();

logToGame(`You engage the ${this.currentMonster.name}!`);

},

attack() {

if (!state.game.combatActive || !this.currentMonster) return;

this.performAction();

},

cast() {

if (!state.game.combatActive || !this.currentMonster) return;

this.performAction();

},

performAction() {

const player = state.player;

const enemy = this.currentMonster;

// GDD FORMULA IMPLEMENTATION

const attackClass = player.class === 'Fighter' ? player.stats.finalWC : player.stats.finalSC;

let playerDamage = (90 \* attackClass) / enemy.AC;

playerDamage = Math.max(1, playerDamage);

enemy.hp -= playerDamage;

logToGame(`You strike ${enemy.name} for ${playerDamage.toFixed(0)} damage!`);

if (enemy.hp <= 0) {

enemy.hp = 0;

logToGame(`${enemy.name} defeated!`);

ItemManager.progressShadowItems(); // Progress equipped shadow items on kill

const defeatedMsg = document.getElementById('enemy-defeated-msg');

const monsterName = document.getElementById('monster-name');

const hpBar = document.getElementById('monster-hp-bar').parentElement;

const hpText = document.getElementById('monster-hp-text');

defeatedMsg.style.display = 'block';

monsterName.style.display = 'none';

hpBar.style.display = 'none';

hpText.style.display = 'none';

let xpReward = enemy.xpValue;

let goldReward = enemy.goldValue;

if (enemy.isBoss) {

if (!player.defeatedBosses.includes(enemy.zoneId)) {

// First-kill milestone

xpReward \*= 10;

goldReward \*= 1000;

player.defeatedBosses.push(enemy.zoneId);

showToast(`First kill bonus! Zone tier unlocked!`, false);

TeleportManager.populateZoneList();

} else {

// Subsequent kills

xpReward \*= 1.5;

goldReward \*= 10;

}

}

ProfileManager.addXp(xpReward);

ProfileManager.addGold(goldReward);

const loot = ItemManager.generateAndAwardLoot(state.game.currentZoneTier, enemy.guaranteedDrop);

this.updateCombatInfoPanel(loot);

this.endCombat();

} else {

// GDD FORMULA IMPLEMENTATION

let enemyDamage = enemy.attack - (player.stats.finalAC \* 0.5);

enemyDamage = Math.max(1, enemyDamage);

player.hp -= enemyDamage;

logToGame(`${enemy.name} strikes you for ${enemyDamage.toFixed(0)} damage!`);

if (player.hp <= 0) {

player.hp = 0;

logToGame("You have been defeated! You are returned to the sanctuary.");

ProfileManager.healPlayer();

WorldMapManager.playerPos = { q: 0, r: 0 };

WorldMapManager.draw();

WorldMapManager.updateInteractButton();

this.endCombat();

}

this.updateEnemyUI();

}

ProfileManager.updateAllProfileUI();

},

endCombat() {

state.game.combatActive = false;

this.updateButtons();

},

updateEnemyUI() {

const monsterHpText = document.getElementById('monster-hp-text');

const monsterHpBar = document.getElementById('monster-hp-bar');

if (this.currentMonster && monsterHpText && monsterHpBar) {

const hpPercent = (this.currentMonster.hp / this.currentMonster.maxHp) \* 100;

monsterHpBar.style.width = `${hpPercent}%`;

monsterHpText.textContent = `Enemy Health: ${Math.ceil(this.currentMonster.hp)} / ${Math.ceil(this.currentMonster.maxHp)}`;

}

},

updateCombatInfoPanel(loot = {}) {

const combatDrop = document.getElementById('last-drop-value');

const combatGem = document.getElementById('gem-drop-value');

if (combatDrop) combatDrop.textContent = loot.item ? loot.item.name : 'None';

if (combatGem) combatGem.textContent = loot.gem ? loot.gem.name : 'None';

},

updateButtons() {

const engageBtn = ui.tabContentCombat.querySelector('#engageBtn');

const attackBtn = ui.tabContentCombat.querySelector('#attackBtn');

const castBtn = ui.tabContentCombat.querySelector('#castBtn');

if (!state.player.class) return;

const type = state.player.class;

engageBtn.disabled = !this.currentMonster || state.game.combatActive;

const canAct = this.currentMonster && state.game.combatActive;

attackBtn.disabled = !canAct;

castBtn.disabled = !canAct;

if (type === 'Fighter') {

attackBtn.style.display = 'block';

castBtn.style.display = 'none';

} else {

attackBtn.style.display = 'none';

castBtn.style.display = 'block';

}

}

};

const ItemManager = {

generateAndAwardLoot(zoneTier, guaranteedDrop = null) {

let itemDropped = null;

let gemDropped = null;

const baseShadowChance = 1 / 600;

const baseGemChance = 1 / 250;

// Handle guaranteed drops from special monsters

if (guaranteedDrop === 'Shadow') {

itemDropped = this.createShadowDrop(zoneTier);

} else if (guaranteedDrop === 'Gem') {

gemDropped = this.createGemDrop();

}

// Regular drop chances if no guaranteed drop occurred

if (!itemDropped && Math.random() < baseShadowChance) {

itemDropped = this.createShadowDrop(zoneTier);

}

if (!gemDropped && Math.random() < baseGemChance) {

gemDropped = this.createGemDrop();

}

EquipmentManager.updateAllViews();

return { item: itemDropped, gem: gemDropped };

},

createShadowDrop(zoneTier) {

const p = state.player;

const equippedDroppers = Object.values(p.equipment)

.map(id => p.inventory.find(i => i.instanceId === id))

.filter(item => item && item.type === 'Dropper');

if (equippedDroppers.length === 0) {

return null; // No droppers equipped, no shadow can drop.

}

const itemToShadow = equippedDroppers[Math.floor(Math.random() \* equippedDroppers.length)];

// Check if a shadow of this item is already equipped

const isShadowEquipped = Object.values(p.equipment)

.map(id => p.inventory.find(i => i.instanceId === id))

.some(item => item && item.type === 'Shadow' && item.baseItemId === itemToShadow.baseItemId);

if (isShadowEquipped) {

showToast("An Echo was formed!", false);

console.log(`Echo created for ${itemToShadow.baseItemId}`);

return { name: "Echo Fragment", type: "Echo"};

} else {

const newItem = GameData.ItemFactory.createItemInstance(itemToShadow.baseItemId, zoneTier, 'Shadow');

if (newItem) {

p.inventory.push(newItem);

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === newItem.baseItemId);

showToast(`Shadow Drop: ${baseItem.name}!`, false);

return { name: `Shadow ${baseItem.name}`, type: 'Shadow' };

}

}

return null;

},

createGemDrop() {

const gemIds = Object.keys(GameData.Gems);

const randomGemId = gemIds[Math.floor(Math.random() \* gemIds.length)];

state.player.gems.push({ id: randomGemId, grade: 1 });

const gemData = GameData.Gems[randomGemId];

showToast(`Gem Drop: ${gemData.name}!`, false);

return gemData;

},

progressShadowItems() {

const KILLS\_PER\_QUALITY\_POINT = 100; // 100 kills to increase quality by 0.01 (1%)

let statsChanged = false;

Object.values(state.player.equipment).forEach(instanceId => {

if (!instanceId) return;

const item = state.player.inventory.find(i => i.instanceId === instanceId);

if (item && item.type === 'Shadow' && item.quality < 1.5) {

item.kills = (item.kills || 0) + 1;

if (item.kills % KILLS\_PER\_QUALITY\_POINT === 0) {

item.quality = Math.min(1.5, parseFloat((item.quality + 0.01).toFixed(2)));

// Recalculate classValue based on new quality

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

const baseClassValue = 13 \* Math.pow(1.22, item.tier - 1);

item.classValue = (baseClassValue \* baseItem.proportion) \* item.quality;

showToast(`${baseItem.name}'s quality improved to ${(item.quality \* 100).toFixed(0)}%!`, false);

statsChanged = true;

}

}

});

if (statsChanged) {

ProfileManager.calculateAllStats();

}

}

};

const WorldMapManager = { isInitialized: false, grid: new Map(), playerPos: { q: 0, r: 0 }, hexSize: 18, ctx: null, init() { if (this.isInitialized) return; this.isInitialized = true; ui.miniMapCanvas.width = ui.miniMapContainer.clientWidth \* 2; ui.miniMapCanvas.height = ui.miniMapContainer.clientHeight \* 2; ui.miniMapCanvas.style.width = `${ui.miniMapContainer.clientWidth}px`; ui.miniMapCanvas.style.height = `${ui.miniMapContainer.clientHeight}px`; this.ctx = ui.miniMapCanvas.getContext('2d'); this.generateGrid(); this.updateInteractButton(); this.draw(); }, generateGrid() { for (let q = -3; q <= 3; q++) { for (let r = -3; r <= 3; r++) { const s = -q - r; if (s >= -3 && s <= 3) { this.grid.set(`${q},${r}`, { q, r, s, feature: { name: 'Monster Zone', icon: '◻️' } }); } } } this.grid.get('1,-1').feature = { name: 'Weapons/Combat Shop', icon: '⚔️' }; this.grid.get('-1,1').feature = { name: 'Magic/Accessories Shop', icon: '🔮' }; this.grid.get('2,0').feature = { name: 'Bank', icon: '🏧' }; this.grid.get('-2,0').feature = { name: 'Sanctuary', icon: '🆘' }; this.grid.get('0,2').feature = { name: 'Teleport Zone', icon: '🌀' }; }, movePlayer(dq, dr) { const newQ = this.playerPos.q + dq; const newR = this.playerPos.r + dr; if (this.grid.has(`${newQ},${newR}`)) { this.playerPos.q = newQ; this.playerPos.r = newR; this.draw(); this.updateInteractButton(); const currentHex = this.grid.get(`${this.playerPos.q},${this.playerPos.r}`); if (currentHex && currentHex.feature && currentHex.feature.name === 'Monster Zone') { CombatManager.populateMonsterList(state.game.currentZoneTier); } else { CombatManager.clearMonsterList(); } } }, updateInteractButton() { const currentHex = this.grid.get(`${this.playerPos.q},${this.playerPos.r}`); const interactKey = document.getElementById('key-interact'); if (currentHex && currentHex.feature && currentHex.feature.name !== 'Monster Zone') { interactKey.textContent = `Enter`; interactKey.style.fontSize = '14px'; } else { interactKey.textContent = 'Interact'; interactKey.style.fontSize = '16px'; } }, handleInteraction() { const currentHex = this.grid.get(`${this.playerPos.q},${this.playerPos.r}`); if (currentHex && currentHex.feature) { if (currentHex.feature.name === 'Weapons/Combat Shop') { ShopManager.openShop('armory'); } else if (currentHex.feature.name === 'Magic/Accessories Shop') { ShopManager.openShop('magic'); } else if (currentHex.feature.name === 'Bank') { BankManager.openBank(); } else if (currentHex.feature.name === 'Sanctuary') { ProfileManager.healPlayer(); } else if (currentHex.feature.name === 'Teleport Zone') { TeleportManager.showModal(); } } }, draw() { const canvas = ui.miniMapCanvas; this.ctx.clearRect(0, 0, canvas.width, canvas.height); const centerX = canvas.width / 2; const centerY = canvas.height / 2; this.grid.forEach(hex => { const relQ = hex.q - this.playerPos.q; const relR = hex.r - this.playerPos.r; const {x, y} = HexUtils.hexToPixel(relQ, relR, this.hexSize); this.drawHex(centerX + x, centerY + y, this.hexSize, hex.feature); }); this.drawPlayer(centerX, centerY); }, drawHex(cx, cy, size, feature) { this.ctx.beginPath(); for (let i = 0; i < 6; i++) { const angle = 2 \* Math.PI / 6 \* (i + 0.5); const x = cx + size \* Math.cos(angle); const y = cy + size \* Math.sin(angle); if (i === 0) this.ctx.moveTo(x, y); else this.ctx.lineTo(x, y); } this.ctx.closePath(); this.ctx.fillStyle = 'rgba(10, 10, 10, 0.5)'; this.ctx.fill(); this.ctx.strokeStyle = 'rgba(249, 115, 22, 0.3)'; this.ctx.lineWidth = 1.5; this.ctx.stroke(); if (feature) { this.ctx.font = `${size \* 1.5}px sans-serif`; this.ctx.textAlign = 'center'; this.ctx.textBaseline = 'middle'; this.ctx.fillText(feature.icon, cx, cy); } }, drawPlayer(cx, cy) { this.ctx.font = `${this.hexSize \* 1.5}px sans-serif`; this.ctx.textAlign = 'center'; this.ctx.textBaseline = 'middle'; this.ctx.fillText('🟠', cx, cy); } };

const BankManager = { isInitialized: false, init() { if (this.isInitialized) return; this.isInitialized = true; }, openBank() { this.renderBankUI(); }, renderBankUI() { const contentHTML = ` <div id="bank-content" class="p-4 text-center"> <div class="grid grid-cols-2 gap-4 mb-4 text-lg"> <div> <div class="text-sm text-gray-400 font-orbitron">Your Gold</div> <div id="bank-player-gold" class="font-bold text-yellow-400 font-orbitron">${state.player.gold.toLocaleString()}</div> </div> <div> <div class="text-sm text-gray-400 font-orbitron">Banked Gold</div> <div id="bank-vault-gold" class="font-bold text-yellow-400 font-orbitron">${state.player.bankGold.toLocaleString()}</div> </div> </div> <input type="number" id="bank-amount-input" class="w-full p-2 rounded text-lg text-black bg-gray-200" placeholder="Enter amount..."> <div class="grid grid-cols-2 gap-2 mt-4"> <button id="bank-deposit-btn" class="glass-button py-2 rounded-md">Deposit</button> <button id="bank-withdraw-btn" class="glass-button py-2 rounded-md">Withdraw</button> </div> </div> `; ModalManager.show('Bank Vault', contentHTML, { onContentReady: (contentDiv) => { contentDiv.querySelector('#bank-deposit-btn').addEventListener('click', () => this.handleTransaction('deposit')); contentDiv.querySelector('#bank-withdraw-btn').addEventListener('click', () => this.handleTransaction('withdraw')); } }); }, handleTransaction(type) { const input = document.getElementById('bank-amount-input'); const amount = parseInt(input.value); if (isNaN(amount) || amount <= 0) { showToast("Please enter a valid amount.", true); return; } if (type === 'deposit') { if (amount > state.player.gold) { showToast("You don't have enough gold to deposit.", true); return; } state.player.gold -= amount; state.player.bankGold += amount; showToast(`Deposited ${amount.toLocaleString()} gold.`); } else if (type === 'withdraw') { if (amount > state.player.bankGold) { showToast("You don't have enough gold in the bank.", true); return; } state.player.bankGold -= amount; state.player.gold += amount; showToast(`Withdrew ${amount.toLocaleString()} gold.`); } input.value = ''; ProfileManager.updateAllProfileUI(); document.getElementById('bank-player-gold').textContent = state.player.gold.toLocaleString(); document.getElementById('bank-vault-gold').textContent = state.player.bankGold.toLocaleString(); } };

const ShopManager = { isInitialized: false, init() { if (this.isInitialized) return; this.isInitialized = true; }, openShop(shopType) { const contentHTML = ` <div class="p-4 text-center"> <h3 class="font-orbitron text-lg mb-2">Welcome to the ${shopType} Shop!</h3> <p class="text-gray-400">Shop functionality is not yet implemented.</p> <div class="mt-4"> <div class="shop-item-row"> <span>Item Name</span> <span>Description</span> <span>Price</span> </div> <div class="shop-item-row"> <span>Placeholder Item</span> <span>A nice placeholder.</span> <span class="text-yellow-400">100g</span> </div> </div> </div> `; ModalManager.show(`${shopType.charAt(0).toUpperCase() + shopType.slice(1)} Shop`, contentHTML); } };

const TeleportManager = {

isInitialized: false,

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.populateZoneList();

this.addEventListeners();

},

addEventListeners() {

ui.zoneTeleportTrigger.addEventListener('click', () => {

this.showModal();

});

ui.zonePopupClose.addEventListener('click', () => {

this.hideModal();

});

ui.zonePopupBackdrop.addEventListener('click', () => {

this.hideModal();

});

ui.zoneListContainer.addEventListener('click', (e) => {

const item = e.target.closest('li');

if (item && !item.classList.contains('disabled')) {

this.handleTeleport(item.dataset.zoneId);

}

});

},

showModal() {

this.populateZoneList();

ui.zonePopupModal.classList.remove('hidden');

},

hideModal() {

ui.zonePopupModal.classList.add('hidden');

},

populateZoneList() {

const container = ui.zoneListContainer;

if (!container || !state.player.level || Object.keys(AllZones).length === 0) return;

const zones = Object.entries(AllZones)

.map(([id, zone]) => ({ id, ...zone }))

.sort((a, b) => a.levelReq - b.levelReq);

container.innerHTML = zones.map(zone => {

const isUnlocked = state.player.level >= zone.levelReq;

return `

<li class="${!isUnlocked ? 'disabled' : ''} font-orbitron" data-zone-id="${zone.id}" title="${isUnlocked ? zone.name : `Requires Level ${zone.levelReq}`}">

${zone.name} - Level ${zone.levelReq}

</li>`;

}).join('');

},

handleTeleport(zoneId) {

if (!zoneId) return;

const zone = AllZones[zoneId];

if (!zone) return;

state.game.currentZoneTier = parseInt(zoneId);

WorldMapManager.playerPos = { q: 0, r: 0 };

WorldMapManager.draw();

WorldMapManager.updateInteractButton();

const landingHex = WorldMapManager.grid.get(`${WorldMapManager.playerPos.q},${WorldMapManager.playerPos.r}`);

if (landingHex && landingHex.feature && landingHex.feature.name === 'Monster Zone') {

CombatManager.populateMonsterList(state.game.currentZoneTier);

} else {

CombatManager.clearMonsterList();

}

showToast(`Teleported to ${zone.name}.`);

ProfileManager.updateAllProfileUI();

this.hideModal();

}

};

// --- MODAL MANAGER ---

const ModalManager = {

show(title, contentHTML, options = {}) {

const { widthClass = 'w-11/12 max-w-lg', onContentReady } = options;

ui.modalContainer.innerHTML = `

<div class="modal-backdrop">

<div class="glass-panel p-4 rounded-lg flex flex-col ${widthClass}">

<div class="flex-shrink-0 flex justify-between items-center mb-4">

<h3 class="font-orbitron text-xl capitalize">${title}</h3>

<button id="modal-close-btn" class="text-2xl leading-none transition-colors hover:text-[var(--highlight-orange)]">&times;</button>

</div>

<div id="modal-content-body" class="flex-grow overflow-y-auto custom-scrollbar">${contentHTML}</div>

</div>

</div>`;

ui.modalContainer.querySelector('#modal-close-btn').onclick = () => this.hide();

if (onContentReady) onContentReady(ui.modalContainer.querySelector('#modal-content-body'));

},

hide() {

ui.modalContainer.innerHTML = '';

}

};

// --- CHAT MANAGER ---

const ChatManager = {

isInitialized: false,

currentUser: { id: 'user\_01', name: 'JuugBoyTV', avatarUrl: 'https://placehold.co/64x64/1a1a1a/f97316?text=J' },

onlineUsers: [

{ id: 'user\_01', name: 'JuugBoyTV', avatarUrl: 'https://placehold.co/40x40/1a1a1a/f97316?text=J' },

{ id: 'user\_02', name: 'ShadowStrike', avatarUrl: 'https://placehold.co/40x40/1a1a1a/ffffff?text=S' },

{ id: 'user\_03', name: 'CrimsonBlade', avatarUrl: 'https://placehold.co/40x40/1a1a1a/ffffff?text=C' },

{ id: 'user\_04', name: 'ArcaneWiz', avatarUrl: 'https://placehold.co/40x40/1a1a1a/ffffff?text=A' },

],

localMessages: { main: [], sales: [], clan: [] },

activeChannel: 'main',

replyingToMessage: null,

init() {

if(this.isInitialized) return;

this.isInitialized = true;

this.setupEventListeners();

this.populateOnlineUsers();

this.switchChannel('main');

},

toggleSidebar(show) {

if (show) {

ui.sidebar.classList.remove('sidebar-closed');

ui.sidebar.classList.add('sidebar-open');

ui.sidebarOverlay.classList.remove('hidden');

} else {

ui.sidebar.classList.add('sidebar-closed');

ui.sidebar.classList.remove('sidebar-open');

ui.sidebarOverlay.classList.add('hidden');

}

},

setReplyingTo(message) {

this.replyingToMessage = message;

ui.replyUsername.textContent = message.userName;

ui.replyText.textContent = message.text || '...';

ui.replyIndicator.classList.remove('hidden');

ui.messageInput.focus();

},

cancelReply() {

this.replyingToMessage = null;

ui.replyIndicator.classList.add('hidden');

},

renderAllMessages() {

const messages = this.localMessages[this.activeChannel];

ui.chatMessages.innerHTML = '';

ui.footerChatContentWrapper.innerHTML = '';

messages.forEach(msg => {

this.renderModalMessage(msg);

this.renderFooterMessage(msg);

});

ui.chatMessages.scrollTop = ui.chatMessages.scrollHeight;

ui.footerChatContentWrapper.scrollTop = ui.footerChatContentWrapper.scrollHeight;

},

renderModalMessage(message) {

const isCurrentUser = message.userId === this.currentUser.id;

const messageWrapper = document.createElement('div');

messageWrapper.className = `message-wrapper flex items-start gap-3 mb-4 ${isCurrentUser ? 'justify-end' : 'justify-start'}`;

const time = new Date(message.timestamp).toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' });

let replyHtml = '';

if (message.replyTo) {

replyHtml = `<div class="reply-quote"><p class="font-bold" style="color: var(--highlight-orange);">${message.replyTo.userName}</p><p class="text-gray-300">${message.replyTo.text || '...'}</p></div>`;

}

const messageContent = `<div class="flex flex-col ${isCurrentUser ? 'items-end' : 'items-start'}"><div class="chat-bubble ${isCurrentUser ? 'chat-bubble-user' : 'chat-bubble-other'} rounded-lg px-4 py-2 max-w-xs lg:max-w-md"><p class="text-xs font-bold" style="color: ${isCurrentUser ? 'var(--highlight-orange)' : '#ccc'}">${message.userName}</p>${replyHtml}<p class="text-sm break-words mt-1">${message.text}</p></div></div>`;

const avatarImg = `<img src="${message.avatarUrl}" class="avatar self-start" title="${time}" onerror="this.src='https://placehold.co/40x40/1a1a1a/f97316?text=${message.userName ? message.userName[0] : '?'}'">`;

const replyIconHtml = `<div class="reply-icon p-1" data-message-id="${message.id}"><svg class="w-4 h-4 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M3 10h10a8 8 0 018 8v2M3 10l6-6m-6 6l6 6"></path></svg></div>`;

messageWrapper.innerHTML = isCurrentUser ? `${replyIconHtml}${messageContent}${avatarImg}` : `${avatarImg}${messageContent}${replyIconHtml}`;

ui.chatMessages.appendChild(messageWrapper);

},

renderFooterMessage(msg) {

const p = document.createElement('p');

p.className = 'leading-tight break-words';

const userSpan = document.createElement('strong');

userSpan.className = `mr-1 text-gray-300`;

userSpan.textContent = `[${msg.userName}]:`;

p.appendChild(userSpan);

p.append(document.createTextNode(msg.text));

ui.footerChatContentWrapper.appendChild(p);

},

handleSendMessage() {

const modalText = ui.messageInput.value.trim();

const footerText = ui.footerMessageInput.value.trim();

const text = modalText || footerText;

if (!text) return;

const messageData = { id: `msg\_${Date.now()}`, userId: this.currentUser.id, userName: this.currentUser.name, avatarUrl: this.currentUser.avatarUrl, text, timestamp: Date.now() };

if (this.replyingToMessage) {

messageData.replyTo = { messageId: this.replyingToMessage.id, userName: this.replyingToMessage.userName, text: this.replyingToMessage.text };

}

this.localMessages[this.activeChannel].push(messageData);

ui.messageInput.value = '';

ui.footerMessageInput.value = '';

this.cancelReply();

this.renderAllMessages();

},

switchChannel(channelName) {

this.activeChannel = channelName;

document.querySelectorAll('#tabs-container .tab, .footer-tab-button').forEach(t => {

t.classList.toggle('active', t.dataset.channel === channelName);

});

this.renderAllMessages();

},

populateOnlineUsers() {

ui.onlineUsersList.innerHTML = '';

this.onlineUsers.forEach(user => {

const el = document.createElement('div');

el.className = 'flex items-center gap-2 p-2 rounded-md hover:bg-white/10 cursor-pointer';

el.innerHTML = `<img src="${user.avatarUrl}" class="avatar w-6 h-6" onerror="this.src='https://placehold.co/24x24/1a1a1a/ffffff?text=?'"><span class="text-sm text-gray-300 truncate">${user.name}</span>`;

ui.onlineUsersList.appendChild(el);

});

},

setupEventListeners() {

ui.openChatModalBtn.addEventListener('click', () => ui.chatModal.classList.remove('hidden'));

ui.closeChatModalBtn.addEventListener('click', () => ui.chatModal.classList.add('hidden'));

ui.messageForm.addEventListener('submit', (e) => { e.preventDefault(); this.handleSendMessage(); });

ui.footerMessageForm.addEventListener('submit', (e) => { e.preventDefault(); this.handleSendMessage(); });

ui.openSidebarBtn.addEventListener('click', () => this.toggleSidebar(true));

ui.closeSidebarBtn.addEventListener('click', () => this.toggleSidebar(false));

ui.sidebarOverlay.addEventListener('click', () => this.toggleSidebar(false));

ui.cancelReplyBtn.addEventListener('click', () => this.cancelReply());

ui.chatMessages.addEventListener('click', (e) => {

const replyTarget = e.target.closest('.reply-icon');

if (replyTarget) {

const messageToReply = this.localMessages[this.activeChannel].find(m => m.id === replyTarget.dataset.messageId);

if (messageToReply) this.setReplyingTo(messageToReply);

}

});

ui.tabsContainer.addEventListener('click', (e) => {

const tab = e.target.closest('.tab');

if (tab?.dataset.channel) this.switchChannel(tab.dataset.channel);

});

document.querySelectorAll('.footer-tab-button').forEach(btn => {

btn.addEventListener('click', () => {

if (btn.dataset.channel) this.switchChannel(btn.dataset.channel);

});

});

}

};

// --- SETTINGS MANAGER ---

const SettingsManager = {

isInitialized: false,

themes: {

'Molten Core': {

'--highlight-orange': '#f97316',

'--glow-red': '#ef4444',

'--text-primary': '#f0f0f0',

'--text-secondary': '#a0a0a0',

'--panel-bg': 'rgba(20, 20, 22, 0.75)',

'--input-bg': 'rgba(0, 0, 0, 0.4)',

'--border-color-main': 'rgba(249, 115, 22, 0.4)',

'--border-color-pulse': 'rgba(239, 68, 68, 0.7)',

},

'Frostbite': { // Placeholder theme

'--highlight-orange': '#3b82f6', // blue-500

'--glow-red': '#6366f1', // indigo-500

'--text-primary': '#e5e7eb',

'--text-secondary': '#9ca3af',

'--panel-bg': 'rgba(23, 37, 84, 0.75)',

'--input-bg': 'rgba(0, 0, 0, 0.4)',

'--border-color-main': 'rgba(59, 130, 246, 0.4)',

'--border-color-pulse': 'rgba(99, 102, 241, 0.7)',

}

},

currentThemeIndex: 0,

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.render();

this.addEventListeners();

},

render() {

const settingsTab = document.getElementById('tab-content-settings');

if (!settingsTab) return;

settingsTab.innerHTML = `

<div class="p-4 space-y-6">

<div>

<h3 class="font-orbitron text-lg mb-2">Appearance</h3>

<div class="bg-black/20 p-4 rounded-lg flex items-center justify-between">

<div>

<label class="text-gray-400">Current Theme</label>

<p id="current-theme-name" class="font-bold text-lg font-orbitron">${Object.keys(this.themes)[this.currentThemeIndex]}</p>

</div>

<button id="change-theme-btn" class="glass-button px-4 py-2 rounded-md">Change</button>

</div>

</div>

<div>

<h3 class="font-orbitron text-lg mb-2">Developer Tools</h3>

<div class="bg-black/20 p-4 rounded-lg space-y-4">

<div>

<label for="dev-level-input" class="block text-sm font-medium text-gray-400 mb-1">Set Player Level</label>

<div class="flex gap-2">

<input type="number" id="dev-level-input" class="w-full p-2 rounded text-lg bg-gray-700 placeholder-gray-500 focus:outline-none focus:ring-2 focus:ring-[var(--highlight-orange)] text-white" placeholder="Enter level...">

<button id="dev-set-level-btn" class="glass-button px-4 py-2 rounded-md">Set</button>

</div>

</div>

<div>

<button id="open-dev-tools-btn" class="glass-button w-full py-2 rounded-md mt-4">Open Dev Tools</button>

</div>

</div>

</div>

</div>

`;

},

addEventListeners() {

const settingsTab = document.getElementById('tab-content-settings');

if (!settingsTab) return;

settingsTab.addEventListener('click', (e) => {

if (e.target.id === 'change-theme-btn') {

this.cycleTheme();

} else if (e.target.id === 'dev-set-level-btn') {

const levelInput = document.getElementById('dev-level-input');

const newLevel = parseInt(levelInput.value);

if (!isNaN(newLevel) && newLevel > 0) {

this.setPlayerLevel(newLevel);

levelInput.value = '';

} else {

showToast("Please enter a valid level.", true);

}

} else if (e.target.id === 'open-dev-tools-btn') {

gddEditor.open();

}

});

},

setPlayerLevel(newLevel) {

if (!state.player) return;

state.player.level = newLevel;

state.player.xp = 0; // Reset XP for the new level

state.player.xpToNextLevel = Math.floor(200 \* Math.pow(1.12, state.player.level));

// Recalculate everything that depends on level

ProfileManager.calculateAllStats();

state.player.hp = state.player.stats.maxHp; // Heal on level change for convenience

// Update UI and game systems

TeleportManager.populateZoneList();

QuestManager.assignQuests();

ProfileManager.updateAllProfileUI();

showToast(`Dev: Player level set to ${newLevel}.`, false);

},

applyTheme(themeName) {

const theme = this.themes[themeName];

if (!theme) return;

for (const [key, value] of Object.entries(theme)) {

document.documentElement.style.setProperty(key, value);

}

const themeNameEl = document.getElementById('current-theme-name');

if(themeNameEl) themeNameEl.textContent = themeName;

},

cycleTheme() {

const themeNames = Object.keys(this.themes);

this.currentThemeIndex = (this.currentThemeIndex + 1) % themeNames.length;

const newThemeName = themeNames[this.currentThemeIndex];

this.applyTheme(newThemeName);

}

};

// --- UI MANAGER ---

const UIManager = {

MAX\_INVENTORY: 200,

MAX\_GEMS: 200,

init() {

const focusBtn = document.getElementById('focus-mode-btn');

if (focusBtn) {

focusBtn.addEventListener('click', () => {

ui.mainContent.classList.toggle('focused');

focusBtn.querySelector('#focus-icon-expand').classList.toggle('hidden');

focusBtn.querySelector('#focus-icon-collapse').classList.toggle('hidden');

});

}

},

updateHealthColor(currentHealth, maxHealth) {

const healthValueEl = document.getElementById('health-value');

if (!healthValueEl) return;

healthValueEl.classList.remove('status-ok', 'status-warn', 'status-danger');

const percentage = (currentHealth / maxHealth) \* 100;

if (percentage < 20) healthValueEl.classList.add('status-danger');

else if (percentage < 50) healthValueEl.classList.add('status-warn');

},

updateInventoryColor(currentItems, maxItems) {

const invValueEl = document.getElementById('inv-value');

if (!invValueEl) return;

invValueEl.classList.remove('status-ok', 'status-warn', 'status-danger');

const percentage = (currentItems / maxItems) \* 100;

if (percentage > 80) invValueEl.classList.add('status-danger');

else if (percentage > 50) invValueEl.classList.add('status-warn');

else invValueEl.classList.add('status-ok');

},

updatePlayerStatusUI() {

if (!state.player || !state.player.stats) return;

const p = state.player;

ui.playerNameValue.textContent = p.name;

ui.playerLevelValue.textContent = p.level;

ui.hpBar.style.width = `${(p.hp / p.stats.maxHp) \* 100}%`;

ui.playerHealthNumeric.textContent = `${Math.ceil(p.hp)} / ${Math.ceil(p.stats.maxHp)}`;

},

updateCombatStatsUI() {

if (!state.player || !state.player.stats) return;

const p = state.player;

const healthEl = document.getElementById('health-value');

const expEl = document.getElementById('exp-value');

const nextLvlEl = document.getElementById('next-lvl-value');

const levelEl = document.getElementById('level-value');

const goldEl = document.getElementById('gold-value');

const invEl = document.getElementById('inv-value');

const locationEl = document.getElementById('location-value');

if (healthEl) healthEl.textContent = `${Math.ceil(p.hp)} / ${Math.ceil(p.stats.maxHp)}`;

if (expEl) expEl.textContent = p.xp.toLocaleString();

if (nextLvlEl) nextLvlEl.textContent = p.xpToNextLevel.toLocaleString();

if (levelEl) levelEl.textContent = p.level;

if (goldEl) goldEl.textContent = p.gold.toLocaleString();

if (invEl) {

const invCount = p.inventory.length;

const gemCount = p.gems.length;

invEl.textContent = `${invCount}/${this.MAX\_INVENTORY} | ${gemCount}/${this.MAX\_GEMS}`;

}

if (locationEl && AllZones[state.game.currentZoneTier]) {

locationEl.textContent = AllZones[state.game.currentZoneTier].name;

}

this.updateHealthColor(p.hp, p.stats.maxHp);

this.updateInventoryColor(p.inventory.length, this.MAX\_INVENTORY);

},

flashStatUpdate(attr) {

const statValueEl = document.querySelector(`[data-stat-value="${attr}"]`);

const unspentPointsEl = document.getElementById('unspent-points-value');

if (statValueEl) {

statValueEl.classList.add('flash-update');

setTimeout(() => statValueEl.classList.remove('flash-update'), 500);

}

if (unspentPointsEl) {

unspentPointsEl.classList.add('flash-update');

setTimeout(() => unspentPointsEl.classList.remove('flash-update'), 500);

}

}

};

// --- GDD DATA & DEV TOOLS ---

let GDD = {};

const originalGDD = {}; // Store for original data to allow resetting

function initializeGDD() {

// This will be populated from your GDD file. For now, it has the necessary structure.

const gddData = {

CONSTANTS: {

XP\_FORMULA: "200 \* Math.pow(1.12, level)",

PLAYER\_DAMAGE\_FORMULA: "(90 \* CLASS\_POWER) / MONSTER\_AC",

MONSTER\_DAMAGE\_FORMULA: "MONSTER\_ATTACK - (PLAYER\_AC \* 0.5)",

MONSTER\_HP\_SCALING\_FACTOR: "1.20",

MONSTER\_AC\_SCALING\_FACTOR: "1.20",

MONSTER\_ATK\_SCALING\_FACTOR: "1.22",

REWARD\_SCALING\_FACTOR: "1.27",

},

LEVELS: { MAX\_LEVEL: 400000 },

RACES: {},

ZONES: {},

MONSTERS: {},

BASE\_ITEMS: {},

GEAR\_TIERS: {},

GEMS: {},

ENCHANTMENTS: {},

QUESTS: {},

LOOT\_TABLES: {},

};

GDD = JSON.parse(JSON.stringify(gddData));

Object.assign(originalGDD, JSON.parse(JSON.stringify(gddData)));

}

const gddEditor = {

currentTab: 'dashboard',

adminPassword: 'delete', // Change this password

unlockedTabs: {},

pickerCallback: null,

init() {

// Main listeners

document.getElementById('gdd-editor-close-btn').addEventListener('click', () => this.close());

document.getElementById('gdd-editor-save-btn').addEventListener('click', () => this.save());

document.getElementById('gdd-editor-reset-btn').addEventListener('click', () => this.confirmReset());

document.querySelectorAll('.editor-tab-btn').forEach(btn => {

btn.addEventListener('click', () => this.switchTab(btn.dataset.tab));

});

// Event delegation for dynamic content

const editorContent = document.getElementById('gdd-editor-content');

editorContent.addEventListener('click', (e) => {

const button = e.target.closest('[data-action]');

const header = e.target.closest('.editor-accordion-header');

if (button) {

e.stopPropagation();

const action = button.dataset.action;

const key = button.dataset.key;

const accordion = button.closest('.editor-accordion');

if (action === 'clone-item') this.cloneItem(accordion);

// Add more actions here

} else if (header) {

this.handleAccordionToggle(header);

}

});

},

open() {

this.unlockedTabs = {};

document.getElementById('game-data-editor-modal').classList.remove('hidden');

document.getElementById('game-data-editor-modal').classList.add('flex');

this.populateAllTabs();

this.switchTab(this.currentTab || 'dashboard');

},

close() {

document.getElementById('game-data-editor-modal').classList.add('hidden');

},

switchTab(tabId) {

this.currentTab = tabId;

document.querySelectorAll('.editor-tab-btn').forEach(btn => btn.classList.toggle('active', btn.dataset.tab === tabId));

const editorContentContainer = document.getElementById('gdd-editor-content');

editorContentContainer.querySelectorAll('.editor-tab-content').forEach(content => content.classList.remove('active'));

let tabContent = document.getElementById(`editor-tab-${tabId}`);

if (!tabContent) {

tabContent = document.createElement('div');

tabContent.id = `editor-tab-${tabId}`;

tabContent.className = 'editor-tab-content';

editorContentContainer.appendChild(tabContent);

}

if (tabContent) {

const dataKey = tabId.toUpperCase().replace('\_', '');

const data = GDD[dataKey];

if (data) {

tabContent.innerHTML = this.createFormForData(dataKey, data);

} else {

tabContent.innerHTML = `<h3 class="font-orbitron text-xl mb-4">${tabId.replace('\_', ' ').toUpperCase()}</h3><p>No data found for this section.</p>`;

}

tabContent.classList.add('active');

}

},

populateAllTabs() {

// This function is now mostly handled by switchTab on-demand

},

createFormForData(key, data) {

// This is a simplified version. The full logic would generate the complex forms.

let html = `<h3 class="font-orbitron text-xl mb-4">${key}</h3>`;

html += `<div class="bg-black/20 p-4 rounded-lg">`;

html += `<pre class="text-xs custom-scrollbar">${JSON.stringify(data, null, 2)}</pre>`;

html += `</div>`;

return html;

},

handleAccordionToggle(header) {

header.parentElement.classList.toggle('open');

},

save() {

this.setStatus('Save functionality is being built.', false);

// Full save logic will go here

},

confirmReset() {

// Full reset logic will go here

this.setStatus('Reset functionality is being built.', false);

},

setStatus(message, isError = false, duration = 3000) {

const statusEl = document.getElementById('editor-status');

statusEl.textContent = message;

statusEl.style.color = isError ? '#ef4444' : '#eab308';

if (duration > 0) {

setTimeout(() => statusEl.textContent = '', duration);

}

},

// ... Add other helper functions like showPrompt, showConfirmation, etc. here

};

// --- GAME MANAGER ---

const GameManager = {

isInitialized: false,

init() {

if (this.isInitialized) return;

this.isInitialized = true;

ProfileManager.updateAllProfileUI();

UIManager.init();

StatsManager.init();

CombatManager.init();

EquipmentManager.init();

WorldMapManager.init();

ShopManager.init();

BankManager.init();

TeleportManager.init();

ChatManager.init();

initializeGDD(); // Load the GDD structure

gddEditor.init(); // Initialize the dev tools listeners

SettingsManager.init();

QuestManager.init();

InfusionManager.init();

this.initControls();

this.setupEventListeners();

this.switchTab('inventory');

const landingHex = WorldMapManager.grid.get('0,0');

if (landingHex && landingHex.feature && landingHex.feature.name === 'Monster Zone') {

CombatManager.populateMonsterList(state.game.currentZoneTier);

}

},

initControls() {

const keyElements = document.querySelectorAll('.game-key');

const setKeyState = (key, isPressed) => {

state.keyState[key] = isPressed;

const keyElement = document.getElementById(`key-${key}`);

if (keyElement) {

keyElement.classList.toggle('pressed', isPressed);

}

};

keyElements.forEach(element => {

const key = element.dataset.key;

element.addEventListener('touchstart', (e) => { e.preventDefault(); setKeyState(key, true); this.handleKeyPress(key); });

element.addEventListener('touchend', (e) => { e.preventDefault(); setKeyState(key, false); });

element.addEventListener('mousedown', (e) => { e.preventDefault(); setKeyState(key, true); this.handleKeyPress(key); });

element.addEventListener('mouseup', (e) => { e.preventDefault(); setKeyState(key, false); });

element.addEventListener('mouseleave', () => { if (state.keyState[key]) setKeyState(key, false); });

});

},

handleKeyPress(key) {

switch(key) {

case 'up': WorldMapManager.movePlayer(0, -1); break;

case 'down': WorldMapManager.movePlayer(0, 1); break;

case 'left': WorldMapManager.movePlayer(-1, 0); break;

case 'right': WorldMapManager.movePlayer(1, 0); break;

case 'interact': WorldMapManager.handleInteraction(); break;

}

},

setupEventListeners() {

ui.mainTabsContainer.addEventListener('click', (e) => {

if (e.target.classList.contains('main-tab-button')) {

this.switchTab(e.target.dataset.tab);

}

});

},

switchTab(tabName) {

document.querySelectorAll('#main-tabs-container .main-tab-button, #main-tab-content .main-tab-panel').forEach(el => el.classList.remove('active'));

const tabButton = document.querySelector(`.main-tab-button[data-tab="${tabName}"]`);

const tabPanel = document.getElementById(`tab-content-${tabName}`);

if (tabButton) tabButton.classList.add('active');

if (tabPanel) tabPanel.classList.add('active');

}

};

const InfusionManager = {

isInitialized: false,

selectedGemId: null,

itemFilterState: {

category: 'All',

subType: 'All',

tier: 'All',

quality: 'All',

socketed: 'All',

sortBy: 'tier',

order: 'desc'

},

gemFilterState: {

category: 'All',

grade: 'All'

},

gemCategories: {

'Fighter': { ids: ['warstone', 'warheart', 'agilite', 'mightstone', 'vigorite', 'cripplite', 'weakstone', 'debilitate', 'siphilite', 'sapstone', 'syphonite'] },

'Caster': { ids: ['lorestone', 'loreheart', 'mindrite', 'mindstone', 'sagerite', 'dullrite', 'dullstone', 'drowseite', 'drainrite', 'drawstone', 'leechrite'] },

'Utility': { ids: ['obsidian\_heart', 'spike\_core', 'true\_core', 'veil\_core', 'vital\_core', 'blood\_core', 'flame\_core', 'echoing\_core'] },

'Farming': { ids: ['treasure\_core', 'ascend\_core', 'midas\_core', 'masterwork\_core', 'harvester\_core'] },

},

inventoryBags: {

'Weapon Chest': ['Weapon'],

'Bag of Gear': ['Helmet', 'Armor', 'Leggings', 'Boots', 'Gauntlets'],

'Jewelry Box': ['Amulet', 'Ring'],

'Spell Satchel': ['Spellbook'],

},

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.render();

this.addEventListeners();

this.populateAllFilterOptions();

this.refreshUI();

},

refreshUI() {

this.renderItemBags();

this.renderGemPouch();

},

render() {

const infusionTab = document.getElementById('tab-content-infusion');

if (!infusionTab) return;

let bagsHTML = '';

for (const bagName in this.inventoryBags) {

bagsHTML += `

<div class="stat-accordion-item" data-bag-container="${bagName}">

<button class="stat-accordion-header">

<h3>${bagName} <span id="infusion-${bagName.replace(/\s+/g, '-')}-count" class="text-xs text-gray-500 font-sans"></span></h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content !p-2">

<div class="inventory-grid" data-bag-name="${bagName}"></div>

</div>

</div>`;

}

infusionTab.innerHTML = `

<p class="text-xs text-center text-gray-400 mb-2">Select a gem, then tap an item to open the socketing panel.</p>

<div id="infusion-sort-container" class="mb-2">

<div class="stat-accordion-item open">

<button class="stat-accordion-header">

<h3><svg class="w-5 h-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M3 4h13M3 8h9M3 12h9m-9 4h6"></path></svg>Sort & Filter Items</h3>

<svg class="accordion-arrow w-6 h-6" fill="none" viewBox="0 0 24 24" stroke="currentColor"><path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 5l7 7-7 7" /></svg>

</button>

<div class="stat-accordion-content !p-2">

<div class="grid grid-cols-2 md:grid-cols-3 gap-2">

<div><label class="text-xs text-gray-400">Category</label><select id="infusion-filter-category-select" class="editor-input !w-full !text-xs"></select></div>

<div><label class="text-xs text-gray-400">Sub-Type</label><select id="infusion-filter-subtype-select" class="editor-input !w-full !text-xs"></select></div>

<div><label class="text-xs text-gray-400">Tier</label><select id="infusion-filter-tier-select" class="editor-input !w-full !text-xs"></select></div>

<div><label class="text-xs text-gray-400">Quality</label><select id="infusion-filter-quality-select" class="editor-input !w-full !text-xs"></select></div>

<div><label class="text-xs text-gray-400">Socketed</label><select id="infusion-filter-socketed-select" class="editor-input !w-full !text-xs"></select></div>

</div>

<div class="grid grid-cols-2 gap-2 mt-2 border-t border-gray-700 pt-2">

<div><label class="text-xs text-gray-400">Sort By</label><select id="infusion-sort-by-select" class="editor-input !w-full !text-xs"><option value="tier">Tier</option><option value="name">Name</option><option value="type">Type</option></select></div>

<div><label class="text-xs text-gray-400">Order</label><select id="infusion-sort-order-select" class="editor-input !w-full !text-xs"><option value="desc">Descending</option><option value="asc">Ascending</option></select></div>

</div>

</div>

</div>

</div>

<div id="infusion-item-bags-container">${bagsHTML}</div>

<div class="stat-accordion-item open">

<div class="stat-accordion-header justify-between items-center">

<h3>Gem Pouch <span id="infusion-gem-pouch-count" class="text-xs text-gray-500 font-sans"></span></h3>

<div class="flex items-center gap-2">

<select id="infusion-gem-category-filter" class="editor-input !w-auto !text-xs"><option value="All">All Types</option></select>

<select id="infusion-gem-grade-filter" class="editor-input !w-auto !text-xs"><option value="All">All Grades</option></select>

</div>

</div>

<div class="stat-accordion-content !p-2">

<div id="infusion-gem-pouch-grid" class="gem-pouch-grid"></div>

</div>

</div>

`;

},

addEventListeners() {

const infusionTab = document.getElementById('tab-content-infusion');

if (!infusionTab) return;

infusionTab.addEventListener('click', e => {

const itemSlot = e.target.closest('.inventory-slot');

const gemItem = e.target.closest('.gem-item');

const header = e.target.closest('.stat-accordion-header');

if (itemSlot && itemSlot.dataset.instanceId) {

this.showSocketingModal(itemSlot.dataset.instanceId);

} else if (gemItem && gemItem.dataset.gemId) {

this.selectedGemId = this.selectedGemId === gemItem.dataset.gemId ? null : gemItem.dataset.gemId;

this.renderGemPouch();

} else if (header) {

header.parentElement.classList.toggle('open');

}

});

infusionTab.addEventListener('change', e => {

const targetId = e.target.id;

if (targetId.startsWith('infusion-filter-') || targetId.startsWith('infusion-sort-')) {

this.itemFilterState.category = document.getElementById('infusion-filter-category-select').value;

this.itemFilterState.subType = document.getElementById('infusion-filter-subtype-select').value;

this.itemFilterState.tier = document.getElementById('infusion-filter-tier-select').value;

this.itemFilterState.quality = document.getElementById('infusion-filter-quality-select').value;

this.itemFilterState.socketed = document.getElementById('infusion-filter-socketed-select').value;

this.itemFilterState.sortBy = document.getElementById('infusion-sort-by-select').value;

this.itemFilterState.order = document.getElementById('infusion-sort-order-select').value;

if (targetId === 'infusion-filter-category-select') {

this.populateSubTypeFilter();

this.itemFilterState.subType = 'All';

document.getElementById('infusion-filter-subtype-select').value = 'All';

}

this.renderItemBags();

} else if (targetId.startsWith('infusion-gem-')) {

this.gemFilterState.category = document.getElementById('infusion-gem-category-filter').value;

this.gemFilterState.grade = document.getElementById('infusion-gem-grade-filter').value;

this.renderGemPouch();

}

});

},

populateAllFilterOptions() {

// Item Filters

const categorySelect = document.getElementById('infusion-filter-category-select');

categorySelect.innerHTML = ['All', ...Object.keys(this.inventoryBags)].map(c => `<option value="${c}">${c}</option>`).join('');

const tierSelect = document.getElementById('infusion-filter-tier-select');

let tierOptions = '<option value="All">All Tiers</option>';

for (let i = 1; i <= 20; i++) tierOptions += `<option value="${i}">Tier ${i}</option>`;

tierSelect.innerHTML = tierOptions;

const qualitySelect = document.getElementById('infusion-filter-quality-select');

qualitySelect.innerHTML = ['All', 'Dropper', 'Shadow', 'Echo'].map(q => `<option value="${q}">${q}</option>`).join('');

const socketedSelect = document.getElementById('infusion-filter-socketed-select');

socketedSelect.innerHTML = ['All', 'Yes', 'No'].map(s => `<option value="${s}">${s}</option>`).join('');

this.populateSubTypeFilter();

// Gem Filters

const gemCategorySelect = document.getElementById('infusion-gem-category-filter');

gemCategorySelect.innerHTML += Object.keys(this.gemCategories).map(c => `<option value="${c}">${c}</option>`).join('');

const gemGradeSelect = document.getElementById('infusion-gem-grade-filter');

let gradeOptions = '';

for (let i = 1; i <= 9; i++) gradeOptions += `<option value="${i}">Grade ${i}</option>`;

gemGradeSelect.innerHTML += gradeOptions;

},

populateSubTypeFilter() {

const category = document.getElementById('infusion-filter-category-select').value;

const subTypeSelect = document.getElementById('infusion-filter-subtype-select');

let subTypes = new Set();

const itemsToScan = GameData.ItemFactory.baseItemTemplates.filter(item => {

if (category === 'All') return true;

const typesInBag = this.inventoryBags[category];

return typesInBag && typesInBag.includes(item.type);

});

itemsToScan.forEach(item => {

if (item.subType) subTypes.add(item.subType);

else subTypes.add(item.type);

});

let sortedSubTypes = Array.from(subTypes).sort();

subTypeSelect.innerHTML = ['All', ...sortedSubTypes].map(s => `<option value="${s}">${s}</option>`).join('');

},

renderItemBags() {

const socketableItems = state.player.inventory.filter(itemInstance => {

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === itemInstance.baseItemId);

return baseItem && baseItem.sockets > 0;

});

const { category, subType, tier, quality, socketed } = this.itemFilterState;

const filteredItems = socketableItems.filter(item => {

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

if (!base) return false;

let bagCategory = Object.keys(this.inventoryBags).find(key => this.inventoryBags[key].includes(base.type));

if (category !== 'All' && bagCategory !== category) return false;

if (subType !== 'All' && base.subType !== subType && base.type !== subType) return false;

if (tier !== 'All' && item.tier.toString() !== tier) return false;

if (quality !== 'All' && item.type !== quality) return false;

if (socketed === 'Yes' && (!item.socketedGems || item.socketedGems.filter(g => g).length === 0)) return false;

if (socketed === 'No' && item.socketedGems && item.socketedGems.filter(g => g).length > 0) return false;

return true;

});

filteredItems.sort((a, b) => {

const baseA = GameData.ItemFactory.baseItemTemplates.find(item => item.id === a.baseItemId);

const baseB = GameData.ItemFactory.baseItemTemplates.find(item => item.id === b.baseItemId);

let compareA, compareB;

switch (this.itemFilterState.sortBy) {

case 'name': compareA = baseA.name; compareB = baseB.name; break;

case 'type': compareA = baseA.type; compareB = baseB.type; break;

default: compareA = a.tier; compareB = b.tier; break;

}

if (compareA < compareB) return this.itemFilterState.order === 'asc' ? -1 : 1;

if (compareA > compareB) return this.itemFilterState.order === 'asc' ? 1 : -1;

return 0;

});

document.querySelectorAll('#infusion-item-bags-container .inventory-grid').forEach(grid => grid.innerHTML = '');

filteredItems.forEach(item => {

const base = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

for (const bagName in this.inventoryBags) {

if (this.inventoryBags[bagName].includes(base.type)) {

const grid = document.querySelector(`#infusion-item-bags-container .inventory-grid[data-bag-name="${bagName}"]`);

if (grid) {

// Create gem dots HTML

let gemDotsHTML = '';

if (item.socketedGems && item.socketedGems.filter(g => g).length > 0) {

gemDotsHTML = `<div class="gem-dot-container">${item.socketedGems.filter(g => g).map(() => `<div class="gem-dot"></div>`).join('')}</div>`;

}

grid.innerHTML += `

<div class="inventory-slot" data-instance-id="${item.instanceId}">

${gemDotsHTML}

<img src="${base.imageUrl}" onerror="this.onerror=null;this.src='https://placehold.co/60x60/1f2937/ffffff?text=ERR';">

<span class="item-label">T${item.tier}</span>

</div>`;

}

break;

}

}

});

const isAnyFilterActive = category !== 'All' || subType !== 'All' || tier !== 'All' || quality !== 'All' || socketed !== 'All';

document.querySelectorAll('#infusion-item-bags-container .stat-accordion-item[data-bag-container]').forEach(container => {

const bagName = container.dataset.bagContainer;

const grid = container.querySelector(`.inventory-grid[data-bag-name="${bagName}"]`);

if (isAnyFilterActive) {

container.style.display = (grid && grid.children.length > 0) ? 'block' : 'none';

} else {

container.style.display = 'block';

}

});

this.updateCounts(socketableItems);

},

renderGemPouch() {

const gemGrid = document.getElementById('infusion-gem-pouch-grid');

if (!gemGrid) return;

const { category, grade } = this.gemFilterState;

let filteredGems = state.player.gems.filter(gemInfo => {

if (grade !== 'All' && gemInfo.grade.toString() !== grade) return false;

if (category !== 'All' && !this.gemCategories[category].ids.includes(gemInfo.id)) return false;

return true;

});

gemGrid.innerHTML = filteredGems.map(gemInfo => {

const gem = GameData.Gems[gemInfo.id];

const isSelected = gemInfo.id === this.selectedGemId ? 'selected' : '';

return `<div class="gem-item ${isSelected}" data-gem-id="${gemInfo.id}"><img src="${gem.imageUrl}" class="w-10 h-10" onerror="this.onerror=null;this.src='https://placehold.co/40x40/1f2937/ffffff?text=ERR';"><span class="item-label">${gem.abbreviation}${gemInfo.grade}</span></div>`;

}).join('');

const countEl = document.getElementById('infusion-gem-pouch-count');

if(countEl) countEl.textContent = `(${filteredGems.length}/${state.player.gems.length})`;

},

showSocketingModal(instanceId) {

const itemInstance = state.player.inventory.find(i => i.instanceId === instanceId);

if (!itemInstance) return;

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === itemInstance.baseItemId);

const isEquipped = Object.values(state.player.equipment).includes(instanceId);

const socketsHTML = Array(baseItem.sockets).fill(0).map((\_, index) => {

const gemInfo = itemInstance.socketedGems?.[index];

const gemData = gemInfo ? GameData.Gems[gemInfo.id] : null;

return `<div class="infusion-socket-slot ${gemInfo ? 'has-gem' : ''}" data-socket-index="${index}">${gemInfo ? `<img src="${gemData.imageUrl}" title="${gemData.name}" onerror="this.onerror=null;this.src='https://placehold.co/44x44/1f2937/ffffff?text=ERR';"><span class="item-label">${gemData.abbreviation}${gemInfo.grade}</span>` : ''}</div>`;

}).join('');

const equipButtonHTML = !isEquipped ? `<button id="infusion-equip-btn" class="glass-button w-full py-2 rounded-md mt-4">Equip Item</button>` : '';

const modalContent = `

<div class="focused-item-container text-center">

<img src="${baseItem.imageUrl}" class="mx-auto w-24 h-24 object-contain" onerror="this.onerror=null;this.src='https://placehold.co/100x100/1f2937/ffffff?text=ERR';">

<div class="focused-item-details mt-2">

<div class="item-name font-orbitron text-lg">${baseItem.name}</div>

<div class="item-tier text-sm text-gray-400">Tier ${itemInstance.tier} ${itemInstance.type}</div>

</div>

<div class="sockets-container flex justify-center gap-3 mt-4">${socketsHTML}</div>

${equipButtonHTML}

</div>

`;

ModalManager.show('Infuse Item', modalContent, {

widthClass: 'w-11/12 max-w-sm',

onContentReady: (contentDiv) => {

contentDiv.addEventListener('click', e => {

const socketSlot = e.target.closest('.infusion-socket-slot');

const equipBtn = e.target.closest('#infusion-equip-btn');

if (socketSlot) {

this.handleSocketClick(instanceId, parseInt(socketSlot.dataset.socketIndex));

} else if (equipBtn) {

EquipmentManager.equipItem(instanceId);

ModalManager.hide();

}

});

}

});

},

handleSocketClick(instanceId, socketIndex) {

if (!instanceId) return;

const itemInstance = state.player.inventory.find(i => i.instanceId === instanceId);

if (!itemInstance) return;

if (!Array.isArray(itemInstance.socketedGems)) itemInstance.socketedGems = [];

const existingGem = itemInstance.socketedGems[socketIndex];

if (existingGem) {

const [removedGem] = itemInstance.socketedGems.splice(socketIndex, 1, null);

while (itemInstance.socketedGems.length > 0 && itemInstance.socketedGems[itemInstance.socketedGems.length - 1] === null) {

itemInstance.socketedGems.pop();

}

if (removedGem) state.player.gems.push(removedGem);

showToast("Gem unsocketed.", false);

} else {

if (!this.selectedGemId) { showToast("Please select a gem from your pouch first.", true); return; }

const gemIndex = state.player.gems.findIndex(g => g.id === this.selectedGemId);

if (gemIndex === -1) return;

const [gemToSocket] = state.player.gems.splice(gemIndex, 1);

while (itemInstance.socketedGems.length <= socketIndex) { itemInstance.socketedGems.push(null); }

itemInstance.socketedGems[socketIndex] = gemToSocket;

this.selectedGemId = null;

showToast("Item infused successfully!", false);

}

ProfileManager.calculateAllStats();

this.showSocketingModal(instanceId); // Re-open/refresh the modal

this.refreshUI();

},

updateCounts(socketableItems) {

for (const bagName in this.inventoryBags) {

const itemTypesInBag = this.inventoryBags[bagName];

const count = socketableItems.filter(item => {

const baseItem = GameData.ItemFactory.baseItemTemplates.find(b => b.id === item.baseItemId);

return baseItem && itemTypesInBag.includes(baseItem.type);

}).length;

const countEl = document.getElementById(`infusion-${bagName.replace(/\s+/g, '-')}-count`);

if (countEl) countEl.textContent = `(${count})`;

}

},

};

const QuestManager = {

isInitialized: false,

questData: {

"100-250": [

{ id: "q101", name: "Echoes of the Deep", objective: { type: 'hunt', monster: 'Deep Crawler', zoneId: 25 }, rewards: { xp: 4300000, gold: 290000 } },

{ id: "q102", name: "Desert Scourge", objective: { type: 'hunt', monster: 'Dune Strider', zoneId: 26 }, rewards: { xp: 4500000, gold: 310000 } },

{ id: "q103", name: "Spirelands Shard", objective: { type: 'hunt', monster: 'Ice Drake', zoneId: 28 }, rewards: { xp: 1500000, gold: 100000, item: 'random\_t3\_gem' } },

{ id: "q104", name: "Mountain Heart Shadow", objective: { type: 'hunt', monster: 'Geomancer', zoneId: 29 }, rewards: { xp: 1600000, gold: 110000, item: 'random\_t5\_shadow\_helmet' } },

{ id: "q105", name: "The First Hunt", objective: { type: 'hunt', monster: 'Corrupt Elemental', zoneId: 32 }, rewards: { xp: 1000000, gold: 75000, item: 'talisman\_of\_the\_apprentice' } }

],

"251-500": [

{ id: "q251", name: "Boneyard Secrets", objective: { type: 'hunt', monster: 'Skeleton Warrior', zoneId: 34 }, rewards: { xp: 6800000, gold: 451000 } },

],

"501-1000": [

{ id: "q501", name: "Fungal Horrors", objective: { type: 'hunt', monster: 'Spore Shambler', zoneId: 41 }, rewards: { xp: 10500000, gold: 700000 } },

]

},

init() {

if (this.isInitialized) return;

this.isInitialized = true;

this.renderQuestTab();

this.addEventListeners();

},

addEventListeners() {

ui.tabContentQuest.addEventListener('click', e => {

if (e.target.id === 'claim-quest-rewards-btn') {

this.claimRewards();

} else if (e.target.closest('.quest-item')) {

// Simulate completing a quest for demonstration

const questId = e.target.closest('.quest-item').dataset.questId;

this.completeQuest(questId);

}

});

},

renderQuestTab() {

const questTab = ui.tabContentQuest;

questTab.innerHTML = `

<div id="quest-log-container">

<h2 class="font-orbitron text-2xl mb-4">Quest Log</h2>

<p class="text-sm text-gray-400 mb-4">Click a quest to complete it and add its rewards to your streak pool.</p>

<div id="active-quests"></div>

<div id="quest-streak-container"></div>

</div>

`;

this.updateQuestLog();

},

updateQuestLog() {

const p = state.player;

const activeQuestsContainer = document.getElementById('active-quests');

const streakContainer = document.getElementById('quest-streak-container');

if (!p.activeQuests || p.activeQuests.length === 0) {

activeQuestsContainer.innerHTML = '<p class="text-gray-500">No active quests. Level up to find new adventures!</p>';

} else {

activeQuestsContainer.innerHTML = p.activeQuests.map(quest => `

<div class="quest-item cursor-pointer hover:border-[var(--highlight-orange)]" data-quest-id="${quest.id}">

<h3 class="quest-title">${quest.name}</h3>

<p class="quest-objective">Objective: ${quest.objective.type} a ${quest.objective.monster} in ${AllZones[quest.objective.zoneId].name}</p>

<div class="quest-rewards">

Rewards:

<span class="quest-reward-xp">${quest.rewards.xp.toLocaleString()} XP</span>,

<span class="quest-reward-gold">${quest.rewards.gold.toLocaleString()} Gold</span>

${quest.rewards.item ? `, and a special item!` : ''}

</div>

</div>

`).join('');

}

const multiplier = 1 + (p.questStreak \* 0.1);

streakContainer.innerHTML = `

<div class="quest-streak-panel">

<h3 class="font-orbitron text-lg">Quest Streak</h3>

<div class="quest-streak-value">${p.questStreak} (+${p.questStreak \* 10}%)</div>

<div class="mt-2">

<h4 class="font-bold text-gray-300">Reward Pool</h4>

<p class="quest-pool-item"><span class="quest-reward-xp">${Math.floor(p.questPool.xp \* multiplier).toLocaleString()} XP</span></p>

<p class="quest-pool-item"><span class="quest-reward-gold">${Math.floor(p.questPool.gold \* multiplier).toLocaleString()} Gold</span></p>

<p class="quest-pool-item text-gray-400">${p.questPool.items.length} items</p>

</div>

<button id="claim-quest-rewards-btn" class="glass-button w-full py-2 mt-4 rounded-md" ${p.questStreak === 0 ? 'disabled' : ''}>Claim Rewards</button>

</div>

`;

},

assignQuests() {

const level = state.player.level;

let bracket;

if (level >= 100 && level <= 250) bracket = "100-250";

else if (level >= 251 && level <= 500) bracket = "251-500";

else if (level >= 501 && level <= 1000) bracket = "501-1000";

if (bracket && this.questData[bracket]) {

state.player.activeQuests = [...this.questData[bracket]];

} else {

state.player.activeQuests = [];

}

this.updateQuestLog();

},

completeQuest(questId) {

const p = state.player;

const quest = p.activeQuests.find(q => q.id === questId);

if (!quest) return;

p.questStreak++;

p.questPool.xp += quest.rewards.xp;

p.questPool.gold += quest.rewards.gold;

if (quest.rewards.item) {

p.questPool.items.push(quest.rewards.item);

}

// Remove completed quest from active list

p.activeQuests = p.activeQuests.filter(q => q.id !== questId);

showToast(`Quest "${quest.name}" completed! Rewards added to streak pool.`, false);

this.updateQuestLog();

},

claimRewards() {

const p = state.player;

if (p.questStreak === 0) return;

const multiplier = 1 + (p.questStreak \* 0.1);

const finalXp = Math.floor(p.questPool.xp \* multiplier);

const finalGold = Math.floor(p.questPool.gold \* multiplier);

ProfileManager.addXp(finalXp);

ProfileManager.addGold(finalGold);

// In a real game, you'd add the items from p.questPool.items to inventory

// For now, we'll just log it.

console.log("Claimed items:", p.questPool.items);

showToast(`Streak claimed! Gained ${finalXp.toLocaleString()} XP and ${finalGold.toLocaleString()} Gold.`, false);

// Reset streak

p.questStreak = 0;

p.questPool = { xp: 0, gold: 0, items: [] };

this.assignQuests(); // Re-populate the quest list

this.updateQuestLog();

}

};

// --- NEW: DATA LOADING ---

/\*\*

\* Asynchronously fetches and loads all necessary game data from external JSON files.

\* NOTE: For this to work locally, you need to serve the files from a local server.

\* If you just open the HTML file, you may encounter CORS errors.

\*/

async function loadGameData() {

try {

// In a real project, you would have a 'data/zones.json' file.

// For this simulation, we'll define it here.

const zonesData = { "1": { "name": "Crystal Caves (Dwarf)", "levelReq": 1, "biome": "mountain", "gearTier": 1 }, "2": { "name": "Glimmerwood (Elf)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "3": { "name": "The Shifting Maze (Halfling)", "levelReq": 1, "biome": "plains", "gearTier": 1 }, "4": { "name": "Chromatic Badlands (Human)", "levelReq": 1, "biome": "wastes", "gearTier": 1 }, "5": { "name": "Mana Springs (Gnome)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "6": { "name": "Blazefire Wastes (Dragonborn/Demon)", "levelReq": 1, "biome": "wastes", "gearTier": 1 }, "7": { "name": "Shadow Mire (Tiefling)", "levelReq": 1, "biome": "swamp", "gearTier": 1 }, "8": { "name": "Whispering Woods (Hobbit)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "9": { "name": "Ashfall Barrens (Orc)", "levelReq": 1, "biome": "wastes", "gearTier": 1 }, "10": { "name": "Screaming Crags (Troll)", "levelReq": 1, "biome": "mountain", "gearTier": 1 }, "11": { "name": "The Great Vine Labyrinth (Minotaur)", "levelReq": 1, "biome": "jungle", "gearTier": 1 }, "12": { "name": "The Howling Steppes (Centaur)", "levelReq": 1, "biome": "plains", "gearTier": 1 }, "13": { "name": "Cloud Peaks (Griffin/Angel)", "levelReq": 1, "biome": "mountain", "gearTier": 1 }, "14": { "name": "Emberfall Forest (Phoenix)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "15": { "name": "Aetherial Forests (Unicorn)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "16": { "name": "Grimwater Swamps (Baba Yaga)", "levelReq": 1, "biome": "swamp", "gearTier": 1 }, "17": { "name": "Gravefrost Tundra (Draugr)", "levelReq": 1, "biome": "tundra", "gearTier": 1 }, "18": { "name": "The Sunken City of Lumina (Mermaid)", "levelReq": 1, "biome": "coastal", "gearTier": 1 }, "19": { "name": "Gloomwood (Vampire)", "levelReq": 1, "biome": "forest", "gearTier": 1 }, "20": { "name": "Corrupted Jungles (Werewolf)", "levelReq": 1, "biome": "jungle", "gearTier": 1 }, "21": { "name": "Echoing Chasms (Banshee)", "levelReq": 1, "biome": "mountain", "gearTier": 1 }, "22": { "name": "Sunken Ruins (Paladin)", "levelReq": 1, "biome": "coastal", "gearTier": 1 }, "23": { "name": "Blazefire Wastes (Demon)", "levelReq": 1, "biome": "wastes", "gearTier": 1 }, "24": { "name": "Cloud Peaks (Angel)", "levelReq": 1, "biome": "mountain", "gearTier": 1 }, "25": { "name": "Echoing Chasms", "levelReq": 100, "biome": "mountain", "gearTier": 3 }, "26": { "name": "Starfall Deserts", "levelReq": 110, "biome": "wastes", "gearTier": 3 }, "27": { "name": "The Weeping Mire", "levelReq": 121, "biome": "swamp", "gearTier": 3 }, "28": { "name": "Frozen Spirelands", "levelReq": 135, "biome": "tundra", "gearTier": 4 }, "29": { "name": "Living Mountain", "levelReq": 149, "biome": "mountain", "gearTier": 4 }, "30": { "name": "Chrono-Distorted Fields", "levelReq": 166, "biome": "plains", "gearTier": 4 }, "31": { "name": "Whisperwind Peaks", "levelReq": 184, "biome": "mountain", "gearTier": 5 }, "32": { "name": "Corrupted Jungles", "levelReq": 205, "biome": "jungle", "gearTier": 5 }, "33": { "name": "Acidic Fens", "levelReq": 227, "biome": "swamp", "gearTier": 5 }, "34": { "name": "Bone Deserts", "levelReq": 253, "biome": "wastes", "gearTier": 6 }, "35": { "name": "The Maw", "levelReq": 281, "biome": "wastes", "gearTier": 6 }, "36": { "name": "Poisonbloom Meadows", "levelReq": 312, "biome": "plains", "gearTier": 6 }, "37": { "name": "Storm-Wrenched Coast", "levelReq": 347, "biome": "coastal", "gearTier": 6 }, "38": { "name": "The Rusting Wastes", "levelReq": 386, "biome": "wastes", "gearTier": 7 }, "39": { "name": "Webbed Caverns", "levelReq": 429, "biome": "mountain", "gearTier": 7 }, "40": { "name": "The Scarred Peaks", "levelReq": 477, "biome": "mountain", "gearTier": 7 }, "41": { "name": "Fungal Undergrowth", "levelReq": 530, "biome": "forest", "gearTier": 7 }, "42": { "name": "Obsidian Flats", "levelReq": 589, "biome": "wastes", "gearTier": 7 }, "43": { "name": "Quicksand Dunes", "levelReq": 655, "biome": "wastes", "gearTier": 7 }, "44": { "name": "Floating Islands", "levelReq": 728, "biome": "mountain", "gearTier": 7 }, "45": { "name": "Glass Sea", "levelReq": 809, "biome": "coastal", "gearTier": 7 }, "46": { "name": "Upside-Down Forest", "levelReq": 899, "biome": "forest", "gearTier": 7 }, "47": { "name": "Singing Sands", "levelReq": 1000, "biome": "wastes", "gearTier": 7 }, "48": { "name": "Aurora Borealis Caverns", "levelReq": 1000, "biome": "tundra", "gearTier": 7 }, "49": { "name": "Gloom-Shrouded Peaks", "levelReq": 1000, "biome": "mountain", "gearTier": 7 }, "50": { "name": "Sunken Spire City", "levelReq": 1000, "biome": "coastal", "gearTier": 7 }, "51": { "name": "Giant Mushroom Forests", "levelReq": 1000, "biome": "forest", "gearTier": 8 }, "52": { "name": "Living Stone Gardens", "levelReq": 9143, "biome": "plains", "gearTier": 8 }, "53": { "name": "The Whispering Wastes", "levelReq": 17286, "biome": "wastes", "gearTier": 8 }, "54": { "name": "Mirage Deserts", "levelReq": 25429, "biome": "wastes", "gearTier": 8 }, "55": { "name": "Gravity Wells", "levelReq": 33572, "biome": "mountain", "gearTier": 9 }, "56": { "name": "Chromatic Reefs", "levelReq": 41715, "biome": "coastal", "gearTier": 9 }, "57": { "name": "The Endless Bridge", "levelReq": 49858, "biome": "plains", "gearTier": 9 }, "58": { "name": "Sky-Whale Graveyard", "levelReq": 58001, "biome": "mountain", "gearTier": 9 }, "59": { "name": "The Weaving Caves", "levelReq": 66144, "biome": "mountain", "gearTier": 10 }, "60": { "name": "Echoing Valley of the Giants", "levelReq": 74287, "biome": "plains", "gearTier": 10 }, "61": { "name": "The Glimmering Shore", "levelReq": 82430, "biome": "coastal", "gearTier": 10 }, "62": { "name": "The Whispering Canyon", "levelReq": 90573, "biome": "mountain", "gearTier": 10 }, "63": { "name": "Floating River", "levelReq": 98716, "biome": "coastal", "gearTier": 11 }, "64": { "name": "The Cloud Sea", "levelReq": 106859, "biome": "mountain", "gearTier": 11 }, "65": { "name": "Obsidian Monolith Plains", "levelReq": 115002, "biome": "plains", "gearTier": 11 }, "66": { "name": "The Bloodfang Jungle", "levelReq": 123145, "biome": "jungle", "gearTier": 11 }, "67": { "name": "Sunstone Deserts", "levelReq": 131288, "biome": "wastes", "gearTier": 12 }, "68": { "name": "The Whispering Gardens", "levelReq": 139431, "biome": "forest", "gearTier": 12 }, "69": { "name": "Glass Peaks", "levelReq": 147574, "biome": "mountain", "gearTier": 12 }, "70": { "name": "Phantom Forests", "levelReq": 155717, "biome": "forest", "gearTier": 13 }, "71": { "name": "The Shrouded Isles", "levelReq": 163860, "biome": "coastal", "gearTier": 13 }, "72": { "name": "Gravity-Defying Rapids", "levelReq": 172003, "biome": "coastal", "gearTier": 13 }, "73": { "name": "The Azure Depths", "levelReq": 180146, "biome": "coastal", "gearTier": 14 }, "74": { "name": "Crystalline Spires", "levelReq": 188289, "biome": "mountain", "gearTier": 14 }, "75": { "name": "The Void Scar", "levelReq": 196432, "biome": "wastes", "gearTier": 14 }, "76": { "name": "Living Labyrinth", "levelReq": 204575, "biome": "jungle", "gearTier": 15 }, "77": { "name": "The Silent Sands", "levelReq": 212718, "biome": "wastes", "gearTier": 15 }, "78": { "name": "Acoustic Caves", "levelReq": 220861, "biome": "mountain", "gearTier": 15 }, "79": { "name": "The Glittering Grottos", "levelReq": 229004, "biome": "mountain", "gearTier": 16 }, "80": { "name": "Timeworn Badlands", "levelReq": 237147, "biome": "wastes", "gearTier": 16 }, "81": { "name": "The Canopy Kingdom", "levelReq": 245290, "biome": "jungle", "gearTier": 16 }, "82": { "name": "The Sunken Library", "levelReq": 253433, "biome": "coastal", "gearTier": 16 }, "83": { "name": "Chromatic Geysers", "levelReq": 261576, "biome": "plains", "gearTier": 17 }, "84": { "name": "The Whispering City", "levelReq": 269719, "biome": "plains", "gearTier": 17 }, "85": { "name": "The Labyrinthine Mangroves", "levelReq": 277862, "biome": "swamp", "gearTier": 17 }, "86": { "name": "The Frozen Heart of the World", "levelReq": 286005, "biome": "tundra", "gearTier": 17 }, "87": { "name": "Gelatinous Jungles", "levelReq": 294148, "biome": "jungle", "gearTier": 18 }, "88": { "name": "The Petrified Ocean", "levelReq": 302291, "biome": "coastal", "gearTier": 18 }, "89": { "name": "The Symphony Springs", "levelReq": 310434, "biome": "plains", "gearTier": 18 }, "90": { "name": "The Whispering Cliffs", "levelReq": 318577, "biome": "mountain", "gearTier": 18 }, "91": { "name": "The Bioluminescent Bog", "levelReq": 326720, "biome": "swamp", "gearTier": 19 }, "92": { "name": "The Stone Giant's Graveyard", "levelReq": 334863, "biome": "plains", "gearTier": 19 }, "93": { "name": "The Maze of Roots", "levelReq": 343006, "biome": "jungle", "gearTier": 19 }, "94": { "name": "The Endless Plains of Glass", "levelReq": 351149, "biome": "plains", "gearTier": 19 }, "95": { "name": "The Whispering Temple Ruins", "levelReq": 359292, "biome": "jungle", "gearTier": 19 }, "96": { "name": "The Crystal Ocean", "levelReq": 367435, "biome": "coastal", "gearTier": 19 }, "97": { "name": "The Sunken Palace of the Sea King", "levelReq": 375578, "biome": "coastal", "gearTier": 19 }, "98": { "name": "The Land of Shifting Colors", "levelReq": 383721, "biome": "plains", "gearTier": 19 }, "99": { "name": "The Cloud Forest of the Sky Serpents", "levelReq": 391864, "biome": "forest", "gearTier": 19 }, "100": { "name": "The Singing Rivers", "levelReq": 400000, "biome": "plains", "gearTier": 19 }, "101": { "name": "The Echoing Gorge of Lost Souls", "levelReq": 500000, "biome": "mountain", "gearTier": 20 } };

AllZones = zonesData;

console.log("Zone data successfully loaded.");

// In the future, you can add fetches for items.json, monsters.json, etc. here.

} catch (error) {

console.error("Failed to load game data:", error);

showToast("CRITICAL ERROR: Could not load game data. Please refresh.", true);

}

}

// --- MAIN INITIALIZATION ---

/\*\*

\* Main entry point of the application.

\* Ensures data is loaded before initializing the game modules.

\*/

async function main() {

await loadGameData(); // Ensure data is loaded before the game starts

CreationManager.init();

}

document.addEventListener('DOMContentLoaded', main);

</script>

</body>

</html>