<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>

<title>AFK Dungeon Crawler</title>

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

<link href=<https://fonts.googleapis.com/css2?family=Press+Start+2P&family=Inter:wght@400;700&display=swap> rel=”stylesheet”>

<link rel=”stylesheet” href=<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css>>

<style>

Body {

Font-family: ‘Inter’, sans-serif;

Background: #111827;

Color: #d1d5db;

}

.pixel-font {

Font-family: ‘Press Start 2P’, cursive;

}

.container {

Max-width: 600px;

Margin: 0 auto;

Min-height: 100vh;

}

.main-card {

Background-color: #1f2937;

Border: 2px solid #374151;

Box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.btn {

Background-color: #3b82f6;

Color: #fff;

Padding: 0.75rem 1.5rem;

Border-radius: 9999px;

Transition: all 0.2s;

Text-shadow: 1px 1px 2px rgba(0,0,0,0.2);

Font-weight: bold;

}

.btn:hover {

Background-color: #2563eb;

Transform: translateY(-2px);

}

.btn:disabled {

Background-color: #4b5563;

Color: #9ca3af;

Cursor: not-allowed;

Transform: none;

}

.glass-panel {

Background: rgba(31, 41, 55, 0.5);

Backdrop-filter: blur(10px);

Border: 1px solid rgba(55, 65, 81, 0.5);

}

.modal {

Position: fixed;

Top: 0;

Left: 0;

Width: 100%;

Height: 100%;

Background: rgba(0, 0, 0, 0.7);

Display: flex;

Align-items: center;

Justify-content: center;

z-index: 50;

}

.modal-content {

Background: #1f2937;

Border: 2px solid #3b82f6;

Padding: 2rem;

Border-radius: 1rem;

Max-width: 500px;

Width: 90%;

Max-height: 90%;

Overflow-y: auto;

}

.hp-bar-inner {

Background-color: #ef4444;

Transition: width 0.5s;

}

.rarity-common { color: #9ca3af; }

.rarity-rare { color: #3b82f6; }

.rarity-epic { color: #a855f7; }

.rarity-legendary { color: #f59e0b; }

.rarity-mythic { color: #ef4444; }

.rarity-celestial { color: #fecaca; text-shadow: 0 0 5px #fef2f2; }

</style>

<!—Firebase Libraries 🡪

<script type=”module”>

Import { initializeApp } from <https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js>;

Import { getAuth, signInAnonymously, signInWithCustomToken, onAuthStateChanged } from <https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js>;

Import { getFirestore, doc, setDoc, getDoc, onSnapshot, collection, query, where, getDocs } from <https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js>;

Import { setLogLevel } from <https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js>;

// Global variables provided by the Canvas environment

Const appId = typeof \_\_app\_id !== ‘undefined’ ? \_\_app\_id : ‘default-app-id’;

Const firebaseConfig = JSON.parse(typeof \_\_firebase\_config !== ‘undefined’ ? \_\_firebase\_config : ‘{}’);

Const initialAuthToken = typeof \_\_initial\_auth\_token !== ‘undefined’ ? \_\_initial\_auth\_token : null;

// Game state and UI elements

Let db;

Let auth;

Let userId = null;

Let gameData = {};

Let combatInterval = null;

Let autoCrawlInterval = null;

Let leaderboardData = [];

Let dailyQuests = [];

// Game constants

Const LEVEL\_UP\_XP = 100;

Const GEMS\_PER\_LEVEL = 1;

Const PET\_XP\_TO\_LEVEL = 50;

Const MOUNT\_XP\_TO\_LEVEL = 75;

Const CLASS\_SKILL\_COOLDOWN = 10;

Const MOUNT\_SKILL\_COOLDOWN = 20;

Const CLOAK\_SKILL\_COOLDOWN = 30;

Const CLASSES = {

Thief: { name: ‘Thief’, baseStats: { strength: 10, agility: 20, intelligence: 5 }, icon: ‘🗡️’,

Skills: [{ name: ‘Critical Strike’, description: ‘Next attack has 100% chance to be a critical hit.’, cooldown: CLASS\_SKILL\_COOLDOWN }],

Ability: (gameData) => {

Const { character, combat } = gameData;

Combat.isCritical = true;

addLog(‘You use Critical Strike! Your next attack is guaranteed to be a critical hit.’);

}

},

Sorcerer: { name: ‘Sorcerer’, baseStats: { strength: 5, agility: 10, intelligence: 20 }, icon: ‘🧙‍♂️’,

Skills: [{ name: ‘Fireball’, description: ‘Deals massive elemental damage.’, cooldown: CLASS\_SKILL\_COOLDOWN }],

Ability: (gameData) => {

Const { character, combat } = gameData;

Const damage = character.level \* 5 + character.baseStats.intelligence \* 2;

Combat.currentMonster.hp -= damage;

addLog(`You cast a fireball, dealing ${damage} magical damage!`);

}

},

Archer: { name: ‘Archer’, baseStats: { strength: 15, agility: 15, intelligence: 10 }, icon: ‘🏹’,

Skills: [{ name: ‘Multishot’, description: ‘Attacks twice in one turn.’, cooldown: CLASS\_SKILL\_COOLDOWN }],

Ability: (gameData) => {

gameData.combat.attacksThisTurn = 2;

addLog(‘You use Multishot! You will attack twice this turn.’);

}

}

};

Const PETS = {

Cat: { name: ‘Shadow Cat’, description: ‘Sneaky and fast.’, price: 2, stats: { agility: 5 }, icon: ‘🐈’ },

Dog: { name: ‘War Dog’, description: ‘Loyal and strong.’, price: 2, stats: { strength: 5 }, icon: ‘🐕’ },

Owl: { name: ‘Wise Owl’, description: ‘Knowledgeable companion.’, price: 2, stats: { intelligence: 5 }, icon: ‘🦉’ },

Dragon: { name: ‘Mini Dragon’, description: ‘Breathes small fireballs.’, price: 5, stats: { strength: 10, intelligence: 10 }, icon: ‘🐉’ },

Phoenix: { name: ‘Phoenix Fledgling’, description: ‘Heals you in combat.’, price: 5, stats: { strength: 5, intelligence: 15 }, icon: ‘🐦🔥’ },

Goblin: { name: ‘Treasure Goblin’, description: ‘Finds extra gold in dungeons.’, price: 10, stats: { agility: 5, intelligence: 5 }, icon: ‘😈’ },

Golem: { name: ‘Stone Golem’, description: ‘A sturdy tank that protects you.’, price: 10, stats: { strength: 15 }, icon: ‘🗿’ },

Slime: { name: ‘Corrupt Slime’, description: ‘Corrodes enemy armor.’, price: 10, stats: { intelligence: 15 }, icon: ‘🧪’ }

};

Const MOUNTS = {

Dragon: { name: ‘Doom Dragon’, description: ‘Flames of Destruction.’, price: 5, stats: { strength: 20, agility: 10, intelligence: 15 }, icon: ‘🐲’,

Skill: { name: ‘Flaming Roar’, description: ‘Deals damage based on Strength.’, cooldown: MOUNT\_SKILL\_COOLDOWN },

Ability: (gameData) => {

Const { character, combat } = gameData;

Const damage = character.baseStats.strength \* 3;

Combat.currentMonster.hp -= damage;

addLog(‘Your Doom Dragon roars, dealing ‘ + damage + ‘ fire damage!’);

}

},

Tiger: { name: ‘War Tiger’, description: ‘Raging Pounce.’, price: 5, stats: { strength: 15, agility: 20, intelligence: 10 }, icon: ‘🐅’,

Skill: { name: ‘War Cry’, description: ‘Temporarily boosts your Agility.’, cooldown: MOUNT\_SKILL\_COOLDOWN },

Ability: (gameData) => {

Const { character } = gameData;

gameData.buffs.push({ type: ‘tempAgility’, value: 10, expires: Date.now() + 5000 });

addLog(‘Your War Tiger lets out a mighty war cry, boosting your agility!’);

}

},

Wolf: { name: ‘Chaos Wolf’, description: ‘Shadow Strike.’, price: 5, stats: { strength: 10, agility: 15, intelligence: 20 }, icon: ‘🐺’,

Skill: { name: ‘Chaos Bolt’, description: ‘A magical strike based on Intelligence.’, cooldown: MOUNT\_SKILL\_COOLDOWN },

Ability: (gameData) => {

Const { character, combat } = gameData;

Const damage = character.baseStats.intelligence \* 3;

Combat.currentMonster.hp -= damage;

addLog(‘Your Chaos Wolf fires a Chaos Bolt, dealing ‘ + damage + ‘ magical damage!’);

}

},

Gryphon: { name: ‘Storm Gryphon’, description: ‘Flies you through dungeons.’, price: 15, stats: { strength: 10, agility: 30, intelligence: 10 }, icon: ‘🦅’ },

Direwolf: { name: ‘Dire Wolf’, description: ‘Ferocious and loyal.’, price: 15, stats: { strength: 25, agility: 15, intelligence: 5 }, icon: ‘🐺’ },

Pegasus: { name: ‘Celestial Pegasus’, description: ‘A divine steed.’, price: 30, stats: { strength: 20, agility: 20, intelligence: 20 }, icon: ‘🦄’ },

Elemental: { name: ‘Earth Elemental’, description: ‘Unstoppable force.’, price: 30, stats: { strength: 40, agility: 5, intelligence: 5 }, icon: ‘🌍’ }

};

Const CLOAKS = {

Wind: { name: ‘Cloak of the Zephyr’, stats: { agility: 10 }, rarity: ‘Rare’, price: 10,

Skill: { name: ‘Wind Step’, description: ‘Dodge the next enemy attack.’, cooldown: CLOAK\_SKILL\_COOLDOWN },

Ability: (gameData) => {

gameData.combat.isDodging = true;

addLog(‘You use Wind Step, preparing to dodge the next attack!’);

}

},

Shadow: { name: ‘Cloak of the Shadow’, stats: { intelligence: 10 }, rarity: ‘Rare’, price: 10,

Skill: { name: ‘Shadow Burst’, description: ‘Temporarily increase your Intelligence.’, cooldown: CLOAK\_SKILL\_COOLDOWN },

Ability: (gameData) => {

gameData.buffs.push({ type: ‘tempInt’, value: 15, expires: Date.now() + 5000 });

addLog(‘You use Shadow Burst, increasing your magical prowess!’);

}

},

Titan: { name: ‘Mantle of the Titan’, stats: { strength: 15 }, rarity: ‘Epic’, price: 20,

Skill: { name: ‘Titan Slam’, description: ‘Stun the enemy for one turn.’, cooldown: CLOAK\_SKILL\_COOLDOWN },

Ability: (gameData) => {

gameData.combat.isStunned = true;

addLog(‘You use Titan Slam, stunning the monster!’);

}

},

Vampire: { name: ‘Cloak of the Leech’, stats: { strength: 10, agility: 10, intelligence: 10 }, rarity: ‘Legendary’, price: 40,

Skill: { name: ‘Vampiric Touch’, description: ‘Steal HP from the enemy.’, cooldown: CLOAK\_SKILL\_COOLDOWN },

Ability: (gameData) => {

Const damage = gameData.character.level \* 3;

Const heal = damage / 2;

gameData.combat.currentMonster.hp -= damage;

gameData.character.hp = Math.min(gameData.character.maxHp, gameData.character.hp + heal);

addLog(`You use Vampiric Touch, stealing ${heal} HP from the monster!`);

}

},

Celestial: { name: ‘Celestial Shroud’, stats: { strength: 20, agility: 20, intelligence: 20 }, rarity: ‘Mythic’, price: 75,

Skill: { name: ‘Divine Light’, description: ‘Heal yourself to full HP.’, cooldown: CLOAK\_SKILL\_COOLDOWN },

Ability: (gameData) => {

gameData.character.hp = gameData.character.maxHp;

addLog(‘You use Divine Light, fully restoring your health!’);

}

}

};

Const SADDLES = {

Leather: { name: ‘Leather Saddle’, bonus: { agility: 5 }, rarity: ‘Common’ },

Steel: { name: ‘Steel Saddle’, bonus: { strength: 5 }, rarity: ‘Rare’ },

Magic: { name: ‘Magic Saddle’, bonus: { intelligence: 5 }, rarity: ‘Epic’ }

};

Const BUFFS = {

xpBoost: { name: ‘XP Boost’, description: ‘Doubles XP gain for 30 minutes.’, price: 5, duration: 1800, effect: { xpMultiplier: 2 } },

goldBoost: { name: ‘Gold Boost’, description: ‘Doubles Gold gain for 30 minutes.’, price: 5, duration: 1800, effect: { goldMultiplier: 2 } },

ultimateBoost: { name: ‘Ultimate Boost’, description: ‘Triples all stats for 1 minute.’, price: 20, duration: 60, effect: { ultimate: 3 } }

};

Const RARITY = {

Common: { name: ‘Common’, color: ‘#9ca3af’, min: 0.0, max: 0.5, next: ‘Rare’ },

Rare: { name: ‘Rare’, color: ‘#3b82f6’, min: 0.5, max: 0.8, next: ‘Epic’ },

Epic: { name: ‘Epic’, color: ‘#a855f7’, min: 0.8, max: 0.95, next: ‘Legendary’ },

Legendary: { name: ‘Legendary’, color: ‘#f59e0b’, min: 0.95, max: 0.99, next: ‘Mythic’ },

Mythic: { name: ‘Mythic’, color: ‘#ef4444’, min: 0.99, max: 1.0, next: null }

};

Const EQUIPMENT\_TYPES = [‘Weapon’, ‘Armor’, ‘Helmet’, ‘Cloak’];

Const STAT\_TYPES = [‘strength’, ‘agility’, ‘intelligence’];

Const INITIAL\_WEAPONS = [

{ name: ‘Rusty Sword’, stats: { strength: 1 }, rarity: ‘Common’ },

{ name: ‘Worn Bow’, stats: { agility: 1 }, rarity: ‘Common’ },

{ name: ‘Cracked Wand’, stats: { intelligence: 1 }, rarity: ‘Common’ }

];

Const CHEST\_RARITIES = [‘Wood’, ‘Gold’, ‘Diamond’, ‘Eternal’];

Const MONSTERS = [

{ name: ‘Goblin’, hp: 20, attack: 5, xp: 10, gold: 5, essence: 1 },

{ name: ‘Slime’, hp: 25, attack: 4, xp: 12, gold: 6, essence: 1 },

{ name: ‘Skeleton’, hp: 30, attack: 7, xp: 15, gold: 7, essence: 1 },

{ name: ‘Orc Warrior’, hp: 40, attack: 9, xp: 20, gold: 10, essence: 2 },

{ name: ‘Dark Imp’, hp: 35, attack: 12, xp: 18, gold: 8, essence: 2 }

];

Const BOSSES = [

{ name: ‘Goblin King’, hp: 100, attack: 15, xp: 50, gold: 25, essence: 5 },

{ name: ‘Slime Lord’, hp: 120, attack: 12, xp: 60, gold: 30, essence: 5 },

{ name: ‘Ancient Golem’, hp: 150, attack: 18, xp: 75, gold: 40, essence: 10 }

];

// UI Element References

Const selectClassScreen = document.getElementById(‘select-class-screen’);

Const gameScreen = document.getElementById(‘game-screen’);

Const mainScreen = document.getElementById(‘main-screen’);

Const dungeonSelectScreen = document.getElementById(‘dungeon-select-screen’);

Const combatScreen = document.getElementById(‘combat-screen’);

Const inventoryScreen = document.getElementById(‘inventory-screen’);

Const skillScreen = document.getElementById(‘skill-screen’);

Const upgradeScreen = document.getElementById(‘upgrade-screen’);

Const petsScreen = document.getElementById(‘pets-screen’);

Const mountsScreen = document.getElementById(‘mounts-screen’);

Const storeScreen = document.getElementById(‘store-screen’);

Const altarScreen = document.getElementById(‘altar-screen’);

Const shrineScreen = document.getElementById(‘shrine-screen’);

Const questScreen = document.getElementById(‘quest-screen’);

Const guildScreen = document.getElementById(‘guild-screen’);

Const potionScreen = document.getElementById(‘potion-screen’);

Const unlocksScreen = document.getElementById(‘unlocks-screen’);

Const characterNameEl = document.getElementById(‘character-name’);

Const characterStatsEl = document.getElementById(‘character-stats’);

Const equipmentEl = document.getElementById(‘equipment’);

Const levelInfoEl = document.getElementById(‘level-info’);

Const goldEl = document.getElementById(‘gold’);

Const gemsEl = document.getElementById(‘gems’);

Const essenceEl = document.getElementById(‘essence’);

Const combatLogEl = document.getElementById(‘combat-log’);

Const userIdDisplay = document.getElementById(‘user-id-display’);

Const playerHpBar = document.getElementById(‘player-hp-bar’);

Const playerHpText = document.getElementById(‘player-hp-text’);

Const monsterHpBar = document.getElementById(‘monster-hp-bar’);

Const monsterHpText = document.getElementById(‘monster-hp-text’);

Const monsterName = document.getElementById(‘monster-name’);

Const playerIcon = document.getElementById(‘player-icon’);

Const playerIconCombat = document.getElementById(‘player-icon-combat’);

Const monsterIcon = document.getElementById(‘monster-icon’);

Const shrineRewardsList = document.getElementById(‘shrine-rewards-list’);

Let activeScreen = ‘main’;

Let isAutoCrawling = false;

Const initialData = (className) => ({

Character: {

Name: `New ${CLASSES[className].name}`,

Class: className,

Level: 1,

Hp: 100,

maxHp: 100,

xp: 0,

skillPoints: 0,

gold: 0,

gems: 0,

essence: 0,

chests: { Wood: 0, Gold: 0, Diamond: 0, Eternal: 0 },

baseStats: { ...CLASSES[className].baseStats, maxHp: 100 },

equipment: { weapon: INITIAL\_WEAPONS[0], armor: null, helmet: null, cloak: null },

inventory: [],

pets: [],

mounts: [],

skills: {},

activeAbilityCooldown: 0,

mountSkillCooldown: 0,

cloakSkillCooldown: 0,

potions: { health: 0, skill: 0 },

},

Dungeon: {

Floor: 1,

enemiesDefeated: 0,

type: ‘Normal’,

difficulty: ‘Easy’

},

Combat: {

currentMonster: null,

combatLog: [],

isCritical: false,

attacksThisTurn: 1,

isDodging: false,

isStunned: false,

},

Buffs: [],

Quests: {

lastReset: Date.now(),

daily: []

},

Leaderboard: []

});

Const changeScreen = (screenName) => {

Const screens = [‘main’, ‘dungeon-select’, ‘combat’, ‘inventory’, ‘skill’, ‘upgrade’, ‘pets’, ‘mounts’, ‘store’, ‘altar’, ‘shrine’, ‘quest’, ‘guild’, ‘potion’, ‘unlocks’];

Screens.forEach(s => {

Document.getElementById(`${s}-screen`).classList.add(‘hidden’);

});

Document.getElementById(`${screenName}-screen`).classList.remove(‘hidden’);

activeScreen = screenName;

if (screenName === ‘guild’) { updateLeaderboards(); }

if (screenName === ‘quest’) { checkDailyQuests(); }

updateUI();

};

Const getRarity = () => {

Const roll = Math.random();

If (roll >= RARITY.Mythic.min) return RARITY.Mythic;

If (roll >= RARITY.Legendary.min) return RARITY.Legendary;

If (roll >= RARITY.Epic.min) return RARITY.Epic;

If (roll >= RARITY.Rare.min) return RARITY.Rare;

Return RARITY.Common;

};

Const generateEquipment = () => {

Const type = EQUIPMENT\_TYPES[Math.floor(Math.random() \* EQUIPMENT\_TYPES.length)];

Const rarity = getRarity();

Const baseStat = STAT\_TYPES[Math.floor(Math.random() \* STAT\_TYPES.length)];

Const levelBonus = gameData.dungeon.floor;

Const difficultyMultiplier = gameData.dungeon.difficulty === ‘Hard’ ? 2 : gameData.dungeon.difficulty === ‘Medium’ ? 1.5 : 1;

Let bonusValue = Math.round(Math.random() \* (RARITY.Mythic.max – RARITY.Mythic.min) \* 10 \* difficultyMultiplier);

bonusValue = Math.max(1, bonusValue);

const newItem = { name: `${rarity.name} ${type}`, stats: { [baseStat]: bonusValue }, rarity: rarity.name, type: type.toLowerCase() };

gameData.character.inventory.push(newItem);

addLog(`Found a new item: <span style=”color:${rarity.color}”>${newItem.name}</span>! It was added to your inventory.`);

};

Window.openChest = (chestType) => {

If (gameData.character.chests[chestType] <= 0) {

addLog(`You have no ${chestType} chests to open.`);

return;

}

Const rarityIndex = CHEST\_RARITIES.indexOf(chestType);

Const rewards = [];

Let goldReward = 100 + rarityIndex \* 500;

Let gemReward = 1 + rarityIndex \* 2;

Let mountSaddle = null;

Let xpBoost = null;

If (chestType === ‘Eternal’) {

mountSaddle = SADDLES[Object.keys(SADDLES)[Math.floor(Math.random() \* Object.keys(SADDLES).length)]];

rewards.push(`${mountSaddle.name} for your mount!`);

if (gameData.character.mounts[0]) {

gameData.character.mounts[0].saddle = mountSaddle;

} else {

addLog(“You found a saddle but you don’t have a mount to put it on!”);

}

}

If (Math.random() < 0.5) {

xpBoost = BUFFS.xpBoost;

rewards.push(`${xpBoost.name}!`);

addBuff(xpBoost.effect.xpMultiplier, ‘xpMultiplier’, xpBoost.duration);

}

gameData.character.gold += goldReward;

gameData.character.gems += gemReward;

rewards.push(`${goldReward} Gold`);

rewards.push(`${gemReward} Gems`);

gameData.character.chests[chestType] -= 1;

updateFirebase();

const message = `You opened a ${chestType} chest and found:\n${rewards.join(‘\n’)}`;

addLog(message);

};

Const addBuff = (value, type, duration) => {

Const existingBuffIndex = gameData.buffs.findIndex(b => b.type === type);

If (existingBuffIndex !== -1) {

gameData.buffs[existingBuffIndex].expires = Date.now() + duration \* 1000;

} else {

gameData.buffs.push({ value, type, expires: Date.now() + duration \* 1000 });

}

};

Const calculateTotalStats = (character) => {

Const totalStats = { ...character.baseStats, hp: character.maxHp, maxHp: character.maxHp };

totalStats.baseHp = character.baseStats.hp;

const applyBonus = (item) => {

if (item && item.stats) {

for (const stat in item.stats) {

totalStats[stat] = (totalStats[stat] || 0) + item.stats[stat];

}

}

};

applyBonus(character.equipment.weapon);

applyBonus(character.equipment.armor);

applyBonus(character.equipment.helmet);

applyBonus(character.equipment.cloak);

character.pets.forEach(pet => {

for (const stat in pet.stats) {

totalStats[stat] = (totalStats[stat] || 0) + pet.stats[stat] \* (pet.level || 1);

}

});

Character.mounts.forEach(mount => {

For (const stat in mount.stats) {

totalStats[stat] = (totalStats[stat] || 0) + mount.stats[stat] \* (mount.level || 1);

}

If (mount.saddle) {

For (const stat in mount.saddle.bonus) {

totalStats[stat] = (totalStats[stat] || 0) + mount.saddle.bonus[stat];

}

}

});

Const tempAgilityBuff = gameData.buffs.find(b => b.type === ‘tempAgility’);

If (tempAgilityBuff) { totalStats.agility += tempAgilityBuff.value; }

Const tempIntBuff = gameData.buffs.find(b => b.type === ‘tempInt’);

If (tempIntBuff) { totalStats.intelligence += tempIntBuff.value; }

Const ultimateBoost = gameData.buffs.find(b => b.type === ‘ultimate’);

If (ultimateBoost) {

For(const stat of STAT\_TYPES) {

totalStats[stat] \*= ultimateBoost.value;

}

}

Return totalStats;

};

Const addLog = (message) => {

Const logEntry = document.createElement(‘p’);

logEntry.innerHTML = message;

combatLogEl.prepend(logEntry);

if (combatLogEl.children.length > 10) {

combatLogEl.removeChild(combatLogEl.lastChild);

}

};

Const updateUI = () => {

If (!gameData.character) return;

Const { character, dungeon, combat } = gameData;

Const totalStats = calculateTotalStats(character);

characterNameEl.textContent = `${character.name} - ${CLASSES[character.class].name}`;

playerIcon.textContent = CLASSES[character.class].icon;

playerIconCombat.textContent = CLASSES[character.class].icon;

let statsHtml = ‘’;

for (const stat in totalStats) {

statsHtml += `<li class=”flex justify-between items-center”><span class=”capitalize font-bold”>${stat}:</span><span>${totalStats[stat]}</span></li>`;

}

characterStatsEl.innerHTML = statsHtml;

let equipmentHtml = ‘’;

for (const type in character.equipment) {

const item = character.equipment[type];

const rarityClass = item?.rarity ? `rarity-${item.rarity.toLowerCase()}` : ‘’;

const statsString = item?.stats ? Object.entries(item.stats).map(([stat, value]) => `${stat}: +${value}`).join(‘, ‘) : ‘None’;

equipmentHtml += `<li class=”flex flex-col”><span class=”font-bold capitalize”>${type}: <span class=”${rarityClass}”>${item?.name || ‘None’}</span></span><span class=”text-xs text-gray-400”>${statsString}</span></li>`;

}

equipmentEl.innerHTML = equipmentHtml;

document.getElementById(‘inventory-list’).innerHTML = gameData.character.inventory.map((item, index) => `

<div class=”flex items-center justify-between p-2 bg-gray-700 rounded-lg”>

<span class=”font-bold rarity-${item.rarity.toLowerCase()}”>${item.name}</span>

<button onclick=”equipItem(${index})” class=”btn text-xs px-2 py-1”>Equip</button>

</div>

`).join(‘’) || ‘<p class=”text-gray-400”>Inventory is empty.</p>’;

Document.getElementById(‘altar-list’).innerHTML = Object.entries(RARITY).slice(0, -1).map(([key, value]) => {

Const itemsOfRarity = gameData.character.inventory.filter(item => item.rarity === key).length;

Const nextRarity = RARITY[key].next;

Const canCombine = itemsOfRarity >= 3 && nextRarity !== null;

Return `

<div class=”flex items-center justify-between p-2 bg-gray-700 rounded-lg”>

<div>

<span class=”font-bold”>${key} Items:</span>

<span class=”rarity-${key.toLowerCase()}”>${itemsOfRarity} / 3</span>

</div>

<button onclick=”combineItems(‘${key}’)” class=”btn text-xs px-2 py-1 ${canCombine ? ‘’ : ‘disabled’}” ${canCombine ? ‘’ : ‘disabled’}>

Combine to ${nextRarity}

</button>

</div>

`;

}).join(‘’) || ‘<p class=”text-gray-400”>You do not have enough items to combine.</p>’;

Document.getElementById(‘pets-list’).innerHTML = gameData.character.pets.map(pet => `

<div class=”flex items-center space-x-2 p-2 bg-gray-700 rounded-lg”>

<span class=”text-2xl”>${pet.icon}</span>

<div class=”flex-grow”>

<span class=”font-bold”>${pet.name} (Lvl ${pet.level})</span>

<div class=”text-sm text-gray-400”>XP: ${pet.xp}/${PET\_XP\_TO\_LEVEL}</div>

</div>

</div>

`).join(‘’) || ‘<p class=”text-gray-400”>You have no pets.</p>’;

Document.getElementById(‘mounts-list’).innerHTML = gameData.character.mounts.map(mount => `

<div class=”flex items-center space-x-2 p-2 bg-gray-700 rounded-lg”>

<span class=”text-2xl”>${mount.icon}</span>

<div class=”flex-grow”>

<span class=”font-bold”>${mount.name} (Lvl ${mount.level})</span>

<div class=”text-sm text-gray-400”>XP: ${mount.xp}/${MOUNT\_XP\_TO\_LEVEL}</div>

${mount.saddle ? `<div class=”text-xs text-yellow-400”>Saddle: ${mount.saddle.name}</div>` : ‘’}

</div>

</div>

`).join(‘’) || ‘<p class=”text-gray-400”>You have no mounts.</p>’;

Document.getElementById(‘skill-list’).innerHTML = CLASSES[character.class].skills.map(skill => `

<div class=”flex items-center justify-between p-2 bg-gray-700 rounded-lg”>

<div class=”flex flex-col”>

<span class=”font-bold”>${skill.name}</span>

<span class=”text-xs text-gray-400”>${skill.description}</span>

</div>

</div>

`).join(‘’);

// Upgrade screen

Document.getElementById(‘skill-points-display’).textContent = `Skill Points: ${gameData.character.skillPoints}`;

Document.getElementById(‘upgrade-buttons’).innerHTML = `

<button onclick=”upgradeStat(‘strength’)” class=”btn px-4 py-3 ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}” ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}>Upgrade Strength (+1)</button>

<button onclick=”upgradeStat(‘agility’)” class=”btn px-4 py-3 ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}” ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}>Upgrade Agility (+1)</button>

<button onclick=”upgradeStat(‘intelligence’)” class=”btn px-4 py-3 ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}” ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}>Upgrade Intelligence (+1)</button>

<button onclick=”upgradeStat(‘maxHp’)” class=”btn px-4 py-3 ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}” ${gameData.character.skillPoints > 0 ? ‘’ : ‘disabled’}>Upgrade Max HP (+10)</button>

`;

levelInfoEl.textContent = `Level: ${character.level} | XP: ${character.xp}/${LEVEL\_UP\_XP}`;

document.getElementById(‘xp-bar-inner’).style.width = `${(character.xp / LEVEL\_UP\_XP) \* 100}%`;

goldEl.textContent = `Gold: ${character.gold}`;

gemsEl.textContent = `Gems: ${character.gems}`;

essenceEl.textContent = `Essence: ${character.essence}`;

// Combat UI

If (combat.currentMonster) {

monsterIcon.textContent = ‘👹’;

monsterName.textContent = combat.currentMonster.name;

monsterHpText.textContent = `HP: ${Math.max(0, combat.currentMonster.hp).toFixed(0)} / ${combat.currentMonster.maxHp}`;

monsterHpBar.style.width = `${(combat.currentMonster.hp / combat.currentMonster.maxHp) \* 100}%`;

playerHpText.textContent = `HP: ${character.hp.toFixed(0)} / ${character.maxHp}`;

playerHpBar.style.width = `${(character.hp / character.maxHp) \* 100}%`;

}

// Button states

Const activeAbilityBtn = document.getElementById(‘active-ability-btn’);

If(activeAbilityBtn){

activeAbilityBtn.disabled = gameData.character.activeAbilityCooldown > 0;

activeAbilityBtn.textContent = gameData.character.activeAbilityCooldown > 0 ? `Ability (${gameData.character.activeAbilityCooldown})` : CLASSES[gameData.character.class].skills[0].name;

}

Const mountAbilityBtn = document.getElementById(‘mount-ability-btn’);

If(mountAbilityBtn){

mountAbilityBtn.disabled = !gameData.character.mounts[0] || gameData.character.mountSkillCooldown > 0;

mountAbilityBtn.textContent = gameData.character.mountSkillCooldown > 0 ? `Mount Skill (${gameData.character.mountSkillCooldown})` : (gameData.character.mounts[0] ? gameData.character.mounts[0].skill.name : ‘No Mount’);

}

Const cloakAbilityBtn = document.getElementById(‘cloak-ability-btn’);

If(cloakAbilityBtn){

cloakAbilityBtn.disabled = !gameData.character.equipment.cloak || gameData.character.cloakSkillCooldown > 0;

cloakAbilityBtn.textContent = gameData.character.cloakSkillCooldown > 0 ? `Cloak (${gameData.character.cloakSkillCooldown})` : (gameData.character.equipment.cloak ? gameData.character.equipment.cloak.skill.name : ‘No Cloak’);

}

// Potion screen

Document.getElementById(‘potion-info’).innerHTML = `

<p>Health Potions: ${gameData.character.potions.health}</p>

<p>Skill Potions: ${gameData.character.potions.skill}</p>

`;

Document.getElementById(‘potion-crafting-buttons’).innerHTML = `

<button onclick=”craftPotion(‘health’)” class=”btn px-4 py-3 ${gameData.character.gold >= 50 && gameData.character.essence >= 5 ? ‘’ : ‘disabled’}” ${gameData.character.gold >= 50 && gameData.character.essence >= 5 ? ‘’ : ‘disabled’}>

Craft Health Potion (50 G, 5 E)

</button>

<button onclick=”craftPotion(‘skill’)” class=”btn px-4 py-3 ${gameData.character.gold >= 100 && gameData.character.essence >= 10 ? ‘’ : ‘disabled’}” ${gameData.character.gold >= 100 && gameData.character.essence >= 10 ? ‘’ : ‘disabled’}>

Craft Skill Potion (100 G, 10 E)

</button>

`;

Document.getElementById(‘potion-use-buttons’).innerHTML = `

<button onclick=”usePotion(‘health’)” class=”btn px-4 py-3 ${gameData.character.potions.health > 0 ? ‘’ : ‘disabled’}” ${gameData.character.potions.health > 0 ? ‘’ : ‘disabled’}>

Use Health Potion

</button>

<button onclick=”usePotion(‘skill’)” class=”btn px-4 py-3 ${gameData.character.potions.skill > 0 ? ‘’ : ‘disabled’}” ${gameData.character.potions.skill > 0 ? ‘’ : ‘disabled’}>

Use Skill Potion

</button>

`;

// Dungeon progress

Document.getElementById(‘dungeon-status’).textContent = `Floor: ${dungeon.floor} | Difficulty: ${dungeon.difficulty} | Monsters: ${dungeon.enemiesDefeated}/10`;

Document.getElementById(‘dungeon-bar-inner’).style.width = `${dungeon.enemiesDefeated \* 10}%`;

// Unlock screen

Const unlocksHtml = `

<li class=”flex items-center space-x-2”><i class=”fas fa-hammer”></i><span>Level 5: Potion Crafting</span></li>

<li class=”flex items-center space-x-2”><i class=”fas fa-scroll”></i><span>Level 5: The Shrine</span></li>

<li class=”flex items-center space-x-2”><i class=”fas fa-hand-sparkles”></i><span>Level 10: Mystic Altar</span></li>

<li class=”flex items-center space-x-2”><i class=”fas fa-users”></i><span>Level 10: Guild Hall (Leaderboards)</span></li>

<li class=”flex items-center space-x-2”><i class=”fas fa-infinity”></i><span>Level 15: Dimensional Rift Dungeon</span></li>

`;

Document.getElementById(‘unlock-list’).innerHTML = unlocksHtml;

// Store UI

updateStoreUI();

};

Const updateStoreUI = () => {

Const petStoreHtml = Object.entries(PETS).map(([id, pet]) => `

<div class=”flex items-center p-4 bg-gray-700 rounded-lg space-x-4 mb-4”>

<span class=”text-4xl flex-shrink-0”>${pet.icon}</span>

<div class=”flex-grow”>

<h4 class=”text-white font-bold text-lg”>${pet.name}</h4>

<p class=”text-sm text-gray-400”>${pet.description}</p>

</div>

<button onclick=”buyItem(‘pet’, ‘${id}’)” class=”btn px-4 py-2 text-sm ${gameData.character.gems >= pet.price ? ‘’ : ‘disabled’}” ${gameData.character.gems >= pet.price ? ‘’ : ‘disabled’}>Buy (${pet.price}💎)</button>

</div>

`).join(‘’);

Document.getElementById(‘pet-store-items’).innerHTML = petStoreHtml;

Const mountStoreHtml = Object.entries(MOUNTS).map(([id, mount]) => `

<div class=”flex items-center p-4 bg-gray-700 rounded-lg space-x-4 mb-4”>

<span class=”text-4xl flex-shrink-0”>${mount.icon}</span>

<div class=”flex-grow”>

<h4 class=”text-white font-bold text-lg”>${mount.name}</h4>

<p class=”text-sm text-gray-400”>${mount.description}</p>

</div>

<button onclick=”buyItem(‘mount’, ‘${id}’)” class=”btn px-4 py-2 text-sm ${gameData.character.gems >= mount.price ? ‘’ : ‘disabled’}” ${gameData.character.gems >= mount.price ? ‘’ : ‘disabled’}>Buy (${mount.price}💎)</button>

</div>

`).join(‘’);

Document.getElementById(‘mount-store-items’).innerHTML = mountStoreHtml;

Const cloakStoreHtml = Object.entries(CLOAKS).map(([id, cloak]) => `

<div class=”flex items-center p-4 bg-gray-700 rounded-lg space-x-4 mb-4”>

<span class=”text-4xl flex-shrink-0”>🥷</span>

<div class=”flex-grow”>

<h4 class=”text-white font-bold text-lg”>${cloak.name}</h4>

<p class=”text-sm text-gray-400”>${cloak.description || ‘A mysterious cloak.’}</p>

</div>

<button onclick=”buyItem(‘cloak’, ‘${id}’)” class=”btn px-4 py-2 text-sm ${gameData.character.gems >= cloak.price ? ‘’ : ‘disabled’}” ${gameData.character.gems >= cloak.price ? ‘’ : ‘disabled’}>Buy (${cloak.price}💎)</button>

</div>

`).join(‘’);

Document.getElementById(‘cloak-store-items’).innerHTML = cloakStoreHtml;

Const premiumStoreHtml = Object.entries(BUFFS).map(([id, buff]) => `

<div class=”flex items-center p-4 bg-gray-700 rounded-lg space-x-4 mb-4”>

<span class=”text-4xl flex-shrink-0”>🚀</span>

<div class=”flex-grow”>

<h4 class=”text-white font-bold text-lg”>${buff.name}</h4>

<p class=”text-sm text-gray-400”>${buff.description}</p>

</div>

<button onclick=”buyItem(‘buff’, ‘${id}’)” class=”btn px-4 py-2 text-sm ${gameData.character.gems >= buff.price ? ‘’ : ‘disabled’}” ${gameData.character.gems >= buff.price ? ‘’ : ‘disabled’}>Buy (${buff.price}💎)</button>

</div>

`).join(‘’);

Document.getElementById(‘premium-store-items’).innerHTML = premiumStoreHtml;

};

Window.equipItem = (index) => {

Const item = gameData.character.inventory[index];

Const currentItem = gameData.character.equipment[item.type];

If (currentItem) {

gameData.character.inventory.push(currentItem);

}

gameData.character.equipment[item.type] = item;

gameData.character.inventory.splice(index, 1);

addLog(`Equipped a new ${item.type}: <span class=”rarity-${item.rarity.toLowerCase()}”>${item.name}</span>!`);

updateFirebase();

updateUI();

};

Window.combineItems = (rarity) => {

Const itemsOfRarity = gameData.character.inventory.filter(item => item.rarity === rarity);

If (itemsOfRarity.length < 3) {

addLog(“Not enough items of this rarity to combine.”);

return;

}

Const nextRarity = RARITY[rarity].next;

If (!nextRarity) {

addLog(“Cannot combine this rarity further.”);

return;

}

For (let i = 0; i < 3; i++) {

Const index = gameData.character.inventory.findIndex(item => item.rarity === rarity);

If (index !== -1) {

gameData.character.inventory.splice(index, 1);

}

}

Const type = EQUIPMENT\_TYPES[Math.floor(Math.random() \* EQUIPMENT\_TYPES.length)];

Const baseStat = STAT\_TYPES[Math.floor(Math.random() \* STAT\_TYPES.length)];

Const rarityData = RARITY[nextRarity];

Let bonusValue = Math.round(Math.random() \* (rarityData.max – rarityData.min) \* 20);

bonusValue = Math.max(1, bonusValue);

const newItem = { name: `${rarityData.name} ${type}`, stats: { [baseStat]: bonusValue }, rarity: rarityData.name, type: type.toLowerCase() };

gameData.character.inventory.push(newItem);

addLog(`Combined 3 ${rarity} items into a new <span class=”rarity-${nextRarity.toLowerCase()}”>${newItem.name}</span>!`);

updateFirebase();

updateUI();

};

Window.buyItem = (itemType, itemId) => {

Const item = (itemType === ‘pet’ ? PETS : itemType === ‘mount’ ? MOUNTS : itemType === ‘cloak’ ? CLOAKS : BUFFS)[itemId];

If (!item) { addLog(‘Item not found.’); return; }

If (gameData.character.gems >= item.price) {

gameData.character.gems -= item.price;

if (itemType === ‘pet’) {

gameData.character.pets.push({ ...item, level: 1, xp: 0 });

} else if (itemType === ‘mount’) {

gameData.character.mounts.push({ ...item, level: 1, xp: 0, saddle: null });

} else if (itemType === ‘cloak’) {

gameData.character.equipment.cloak = { ...item, type: ‘cloak’ };

} else if (itemType === ‘buff’) {

addBuff(item.effect.xpMultiplier || item.effect.goldMultiplier || item.effect.ultimate, Object.keys(item.effect)[0], item.duration);

}

addLog(`You purchased the ${item.name}!`);

updateFirebase();

} else {

addLog(“Not enough gems!”);

}

updateUI();

};

Const levelUp = (target) => {

Const { character } = gameData;

Const targetXp = target === ‘player’ ? LEVEL\_UP\_XP : target === ‘pet’ ? PET\_XP\_TO\_LEVEL : MOUNT\_XP\_TO\_LEVEL;

If (target === ‘player’) {

Character.level += 1;

Character.xp = 0;

Character.skillPoints += 1;

Character.gems += GEMS\_PER\_LEVEL;

addLog(`Level Up! You are now Level ${character.level}! You earned ${GEMS\_PER\_LEVEL} Gem!`);

} else if (target.type === ‘pet’) {

Target.level += 1;

Target.xp = 0;

addLog(`${target.name} leveled up to Lvl ${target.level}!`);

} else if (target.type === ‘mount’) {

Target.level += 1;

Target.xp = 0;

addLog(`${target.name} leveled up to Lvl ${target.level}!`);

}

updateFirebase();

};

Window.upgradeStat = (stat) => {

If (gameData.character.skillPoints > 0) {

gameData.character.skillPoints--;

if (stat === ‘maxHp’) {

gameData.character.baseStats.maxHp += 10;

} else {

gameData.character.baseStats[stat] += 1;

}

addLog(`Upgraded ${stat}!`);

updateFirebase();

updateUI();

}

};

Window.useShrine = () => {

If (gameData.character.gold >= 50 && gameData.character.essence >= 10) {

gameData.character.gold -= 50;

gameData.character.essence -= 10;

const rewards = [];

const roll = Math.random();

if (roll < 0.3) {

const goldBonus = Math.floor(Math.random() \* 500) + 100;

gameData.character.gold += goldBonus;

rewards.push(`+${goldBonus} Gold`);

} else if (roll < 0.6) {

Const gemsBonus = Math.floor(Math.random() \* 5) + 1;

gameData.character.gems += gemsBonus;

rewards.push(`+${gemsBonus} Gems`);

} else if (roll < 0.9) {

Const type = EQUIPMENT\_TYPES[Math.floor(Math.random() \* EQUIPMENT\_TYPES.length)];

Const rarity = getRarity();

Const baseStat = STAT\_TYPES[Math.floor(Math.random() \* STAT\_TYPES.length)];

Let bonusValue = Math.round(Math.random() \* (RARITY.Mythic.max – RARITY.Mythic.min) \* 10 \* 2);

bonusValue = Math.max(1, bonusValue);

const newItem = { name: `${rarity.name} ${type}`, stats: { [baseStat]: bonusValue }, rarity: rarity.name, type: type.toLowerCase() };

gameData.character.inventory.push(newItem);

rewards.push(`a new <span class=”rarity-${rarity.name.toLowerCase()}”>${newItem.name}</span>`);

} else {

Const buff = BUFFS.ultimateBoost;

addBuff(buff.effect.ultimate, ‘ultimate’, buff.duration);

rewards.push(‘an Ultimate Boost!’);

}

shrineRewardsList.innerHTML = rewards.map(r => `<li>- ${r}</li>`).join(‘’);

document.getElementById(‘shrine-result’).classList.remove(‘hidden’);

addLog(`You used the shrine and received: ${rewards.join(‘, ‘)}`);

updateFirebase();

} else {

addLog(“Not enough gold or essence to use the shrine.”);

}

updateUI();

};

Const spawnMonster = () => {

Const { dungeon } = gameData;

Const monsterList = dungeon.enemiesDefeated >= 10 && dungeon.type !== ‘Dimensional’ ? BOSSES : MONSTERS;

Const monster = monsterList[Math.floor(Math.random() \* monsterList.length)];

Const floorModifier = dungeon.floor;

Const scaledMonster = {

...monster,

Hp: monster.hp + (floorModifier \* 10),

maxHp: monster.hp + (floorModifier \* 10),

attack: monster.attack + (floorModifier \* 2),

xp: monster.xp + (floorModifier \* 5),

gold: monster.gold + (floorModifier \* 3),

essence: monster.essence + Math.floor(floorModifier / 5)

};

gameData.combat.currentMonster = scaledMonster;

gameData.combat.combatLog = [];

addLog(`You encountered a ${scaledMonster.name}!`);

updateUI();

};

Const startCombatTick = () => {

If (combatInterval) { clearInterval(combatInterval); }

combatInterval = setInterval(() => {

gameData.character.activeAbilityCooldown = Math.max(0, gameData.character.activeAbilityCooldown – 1);

gameData.character.mountSkillCooldown = Math.max(0, gameData.character.mountSkillCooldown – 1);

gameData.character.cloakSkillCooldown = Math.max(0, gameData.character.cloakSkillCooldown – 1);

checkBuffs();

updateUI();

}, 1000);

};

Window.attack = () => {

Const { character, combat } = gameData;

If (!combat.currentMonster) { return; }

If (combat.isStunned) {

addLog(‘The monster is stunned and cannot attack!’);

combat.isStunned = false;

} else {

Const monsterDamage = Math.max(1, combat.currentMonster.attack);

If (combat.isDodging) {

addLog(‘You dodged the monster\’s attack!’);

combat.isDodging = false;

} else {

Character.hp -= monsterDamage;

addLog(`${combat.currentMonster.name} attacks you for ${monsterDamage} damage.`);

}

}

Const totalStats = calculateTotalStats(character);

Const playerDamage = Math.max(1, totalStats.strength + Math.floor(Math.random() \* totalStats.agility / 2));

Const attacks = combat.attacksThisTurn;

For(let i = 0; i < attacks; i++) {

Const damage = combat.isCritical ? playerDamage \* 2 : playerDamage;

Combat.currentMonster.hp -= damage;

addLog(`You hit ${combat.currentMonster.name} for ${damage} damage. ${combat.isCritical ? ‘<span class=”text-yellow-400”>CRITICAL!</span>’ : ‘’}`);

combat.isCritical = false;

}

gameData.combat.attacksThisTurn = 1;

if (combat.currentMonster.hp <= 0) {

endCombat(true);

} else if (character.hp <= 0) {

endCombat(false);

}

updateUI();

};

Window.useActiveAbility = () => {

Const { character } = gameData;

If (character.activeAbilityCooldown <= 0) {

CLASSES[character.class].ability(gameData);

Character.activeAbilityCooldown = CLASSES[character.class].skills[0].cooldown;

updateFirebase();

updateUI();

}

};

Window.useMountSkill = () => {

Const { character } = gameData;

Const mountAbility = character.mounts[0]?.ability;

If (character.mountSkillCooldown <= 0 && mountAbility) {

mountAbility(gameData);

character.mountSkillCooldown = MOUNTS[Object.keys(MOUNTS)[0]].skill.cooldown;

updateFirebase();

updateUI();

}

};

Window.useCloakAbility = () => {

Const { character } = gameData;

Const cloakAbility = character.equipment.cloak?.ability;

If (character.cloakSkillCooldown <= 0 && cloakAbility) {

cloakAbility(gameData);

character.cloakSkillCooldown = CLOAKS[Object.keys(CLOAKS)[0]].skill.cooldown;

updateFirebase();

updateUI();

}

};

Window.craftPotion = (type) => {

If (type === ‘health’ && gameData.character.gold >= 50 && gameData.character.essence >= 5) {

gameData.character.gold -= 50;

gameData.character.essence -= 5;

gameData.character.potions.health++;

addLog(“Crafted a Health Potion!”);

} else if (type === ‘skill’ && gameData.character.gold >= 100 && gameData.character.essence >= 10) {

gameData.character.gold -= 100;

gameData.character.essence -= 10;

gameData.character.potions.skill++;

addLog(“Crafted a Skill Potion!”);

} else {

addLog(“Not enough resources to craft that potion.”);

}

updateFirebase();

updateUI();

};

Window.usePotion = (type) => {

If (type === ‘health’ && gameData.character.potions.health > 0) {

gameData.character.hp = gameData.character.maxHp;

gameData.character.potions.health--;

addLog(“Used a Health Potion and restored all HP.”);

} else if (type === ‘skill’ && gameData.character.potions.skill > 0) {

gameData.character.activeAbilityCooldown = 0;

gameData.character.mountSkillCooldown = 0;

gameData.character.cloakSkillCooldown = 0;

gameData.character.potions.skill--;

addLog(“Used a Skill Potion and reset all skill cooldowns!”);

} else {

addLog(`You do not have any ${type} potions.`);

}

updateFirebase();

updateUI();

};

Const checkBuffs = () => {

gameData.buffs = gameData.buffs.filter(b => b.expires > Date.now());

updateUI();

};

Window.endCombat = (isVictory) => {

Const { character, combat, dungeon } = gameData;

Const defeatedMonster = combat.currentMonster;

If (isVictory) {

Const xpGain = defeatedMonster.xp \* (gameData.buffs.find(b => b.type === ‘xpMultiplier’)?.value || 1);

Const goldGain = defeatedMonster.gold \* (gameData.buffs.find(b => b.type === ‘goldMultiplier’)?.value || 1);

Const essenceGain = defeatedMonster.essence;

Character.xp += xpGain;

Character.gold += goldGain;

Character.essence += essenceGain;

addLog(`You defeated ${defeatedMonster.name}! Gained ${xpGain} XP, ${goldGain} Gold, and ${essenceGain} Essence.`);

// Pet and Mount XP

Character.pets.forEach(pet => {

Pet.xp += xpGain;

If (pet.xp >= PET\_XP\_TO\_LEVEL) { levelUp({ type: ‘pet’, ...pet }); }

});

Character.mounts.forEach(mount => {

Mount.xp += xpGain;

If (mount.xp >= MOUNT\_XP\_TO\_LEVEL) { levelUp({ type: ‘mount’, ...mount }); }

});

// Dungeon Type Bonuses

If (dungeon.type === ‘Gold’) { character.gold += goldGain \* 2; addLog(`(Gold Dungeon Bonus: +${goldGain \* 2} Gold)`); }

If (dungeon.type === ‘Gem’) { character.gems += 1; addLog(`(Gem Dungeon Bonus: +1 Gem)`); }

If (dungeon.type === ‘Chest’) {

Const rarity = CHEST\_RARITIES[Math.floor(Math.random() \* CHEST\_RARITIES.length)];

Character.chests[rarity] += 1;

addLog(`Found a ${rarity} Chest!`);

}

gameData.dungeon.enemiesDefeated += 1;

if (gameData.dungeon.enemiesDefeated >= 10 && dungeon.type !== ‘Dimensional’) {

gameData.dungeon.floor += 1;

gameData.dungeon.enemiesDefeated = 0;

addLog(`You cleared Floor ${gameData.dungeon.floor – 1}!`);

generateEquipment();

} else if (dungeon.type === ‘Dimensional’) {

gameData.dungeon.floor += 1;

gameData.dungeon.enemiesDefeated = 0;

}

If (character.xp >= LEVEL\_UP\_XP) { levelUp(‘player’); }

// Quest Progress

gameData.quests.daily.forEach(q => {

if (q.type === ‘defeat\_monsters’) q.current += 1;

if (q.type === ‘find\_gold’) q.current += goldGain;

if (q.type === ‘use\_skill’) q.current += 1;

});

} else {

addLog(“You were defeated! Retreating to town...”);

}

Character.hp = character.maxHp;

gameData.combat.currentMonster = null;

updateFirebase();

updateUI();

};

Const startAutoCrawl = () => {

If (isAutoCrawling) {

// If there’s no monster, spawn one

If (!gameData.combat.currentMonster) {

spawnMonster();

}

// Check if combat is over

If (gameData.combat.currentMonster.hp <= 0 || gameData.character.hp <= 0) {

endCombat(gameData.combat.currentMonster.hp <= 0);

// If auto-crawling, immediately start next combat

setTimeout(() => {

if (isAutoCrawling) {

if (gameData.character.hp > 0) { spawnMonster(); }

else {

toggleAutoCrawl();

addLog(“You were defeated! Auto-crawl has stopped.”);

}

}

}, 1000); // Small delay for effect

} else {

// Perform a single attack and use abilities

Attack();

If (gameData.character.activeAbilityCooldown === 0) useActiveAbility();

If (gameData.character.mountSkillCooldown === 0) useMountSkill();

If (gameData.character.equipment.cloak && gameData.character.cloakSkillCooldown === 0) useCloakAbility();

}

updateUI();

}

};

Window.toggleAutoCrawl = () => {

isAutoCrawling = !isAutoCrawling;

document.getElementById(‘auto-crawl-btn’).textContent = isAutoCrawling ? “Stop Auto Crawl” : “Start Auto Crawl”;

document.getElementById(‘auto-crawl-btn’).classList.toggle(‘bg-red-500’, isAutoCrawling);

document.getElementById(‘auto-crawl-btn’).classList.toggle(‘bg-green-500’, !isAutoCrawling);

if (isAutoCrawling) {

addLog(‘Auto Crawl started!’);

if (gameData.dungeon.type === ‘Normal’) {

// Start a default crawl if not already set

gameData.dungeon.type = ‘Normal’;

gameData.dungeon.difficulty = ‘Easy’;

}

Document.getElementById(‘start-dungeon-btn’).disabled = true;

If (!autoCrawlInterval) {

autoCrawlInterval = setInterval(startAutoCrawl, 1000);

startCombatTick();

}

} else {

addLog(‘Auto Crawl stopped.’);

document.getElementById(‘start-dungeon-btn’).disabled = false;

if (autoCrawlInterval) { clearInterval(autoCrawlInterval); autoCrawlInterval = null; }

if (combatInterval) { clearInterval(combatInterval); combatInterval = null; }

endCombat(false);

}

updateFirebase();

updateUI();

};

Window.startDungeon = () => {

If (isAutoCrawling) return;

Const selectedDungeon = document.querySelector(‘input[name=”dungeon-type”]:checked’).value;

Const selectedDifficulty = document.querySelector(‘input[name=”dungeon-difficulty”]:checked’).value;

gameData.dungeon.type = selectedDungeon;

gameData.dungeon.difficulty = selectedDifficulty;

spawnMonster();

changeScreen(‘combat’);

startCombatTick();

};

Const generateQuests = () => {

Const quests = [

{ type: ‘defeat\_monsters’, goal: Math.floor(Math.random() \* 20) + 10, current: 0, reward: { gold: 500, gems: 1, essence: 50 }, description: ‘Defeat monsters.’ },

{ type: ‘find\_gold’, goal: Math.floor(Math.random() \* 1000) + 500, current: 0, reward: { gold: 1000, gems: 2, essence: 100 }, description: ‘Find gold.’ },

{ type: ‘use\_skill’, goal: Math.floor(Math.random() \* 10) + 5, current: 0, reward: { gold: 200, gems: 1, essence: 20 }, description: ‘Use your class skill.’ }

];

Return quests;

};

Const checkDailyQuests = () => {

Const now = Date.now();

Const oneDay = 24 \* 60 \* 60 \* 1000;

If (now – gameData.quests.lastReset > oneDay) {

gameData.quests.daily = generateQuests();

gameData.quests.lastReset = now;

updateFirebase();

}

Const questsHtml = gameData.quests.daily.map((q, index) => {

Const isCompleted = q.current >= q.goal;

Const buttonText = isCompleted ? ‘Claim Reward’ : ‘In Progress’;

Const buttonClasses = `btn px-4 py-2 text-sm ${isCompleted ? ‘’ : ‘disabled’}`;

Return `

<div class=”flex flex-col p-4 bg-gray-700 rounded-lg”>

<h4 class=”text-white font-bold”>${q.description} (${q.current}/${q.goal})</h4>

<p class=”text-sm text-gray-400”>Reward: ${q.reward.gold} Gold, ${q.reward.gems} Gems, ${q.reward.essence} Essence</p>

<button onclick=”claimQuest(${index})” class=”${buttonClasses}” ${isCompleted ? ‘’ : ‘disabled’}>${buttonText}</button>

</div>

`;

}).join(‘’) || ‘<p class=”text-gray-400”>No quests available today.</p>’;

Document.getElementById(‘quest-list’).innerHTML = questsHtml;

};

Window.claimQuest = (index) => {

Const quest = gameData.quests.daily[index];

If (quest && quest.current >= quest.goal) {

gameData.character.gold += quest.reward.gold;

gameData.character.gems += quest.reward.gems;

gameData.character.essence += quest.reward.essence;

gameData.quests.daily.splice(index, 1);

addLog(`Quest completed! Claimed ${quest.reward.gold} Gold, ${quest.reward.gems} Gems, and ${quest.reward.essence} Essence.`);

updateFirebase();

checkDailyQuests();

}

};

Const updateLeaderboards = async () => {

Try {

Const q = query(collection(db, ‘artifacts’, appId, ‘public’, ‘data’, ‘leaderboard’));

Const querySnapshot = await getDocs(q);

leaderboardData = [];

querySnapshot.forEach(doc => {

const data = doc.data();

leaderboardData.push(data);

});

leaderboardData.sort((a, b) => b.highestFloor – a.highestFloor);

let leaderboardHtml = leaderboardData.map((entry, index) => `

<li class=”flex justify-between items-center p-2 bg-gray-700 rounded-lg mb-2”>

<span>${index + 1}. ${entry.name}</span>

<span>Floor: ${entry.highestFloor}</span>

</li>

`).join(‘’);

Document.getElementById(‘leaderboard-list’).innerHTML = leaderboardHtml || ‘<p class=”text-gray-400”>No leaderboard data available.</p>’;

} catch (e) {

Console.error(“Error fetching leaderboard data:”, e);

Document.getElementById(‘leaderboard-list’).innerHTML = ‘<p class=”text-red-400”>Error loading leaderboard.</p>’;

}

};

Const updateFirebase = async () => {

If (!userId) {

Console.error(“User ID is not available. Cannot save data.”);

Return;

}

Try {

Const userDocRef = doc(db, ‘artifacts’, appId, ‘users’, userId, ‘game’, ‘data’);

Await setDoc(userDocRef, JSON.parse(JSON.stringify(gameData)));

// Update public leaderboard

Const leaderboardRef = doc(db, ‘artifacts’, appId, ‘public’, ‘data’, ‘leaderboard’, userId);

Await setDoc(leaderboardRef, {

Name: gameData.character.name,

userId: userId,

highestFloor: gameData.dungeon.floor

}, { merge: true });

} catch (e) {

Console.error(“Error updating game state: “, e);

}

};

Const loadGameData = async (user) => {

userId = user.uid;

userIdDisplay.textContent = `User ID: ${userId}`;

const docRef = doc(db, ‘artifacts’, appId, ‘users’, userId, ‘game’, ‘data’);

onSnapshot(docRef, (docSnap) => {

if (docSnap.exists()) {

gameData = docSnap.data();

gameScreen.classList.remove(‘hidden’);

selectClassScreen.classList.add(‘hidden’);

updateUI();

}

}, (error) => {

Console.error(“Error listening to game data:”, error);

});

};

Const initializeFirebase = async () => {

Try {

Const app = initializeApp(firebaseConfig);

Db = getFirestore(app);

Auth = getAuth(app);

setLogLevel(‘Debug’);

onAuthStateChanged(auth, async (user) => {

if (user) { await loadGameData(user); } else {

if (initialAuthToken) { await signInWithCustomToken(auth, initialAuthToken); } else { await signInAnonymously(auth); }

}

});

} catch (e) {

Console.error(“Error initializing Firebase:”, e);

}

};

Window.onload = initializeFirebase;

// Corrected character selection logic to ensure immediate transition

Window.selectClass = async (className) => {

gameData = initialData(className);

await updateFirebase();

changeScreen(‘main’);

};

// Navigation Buttons

Document.getElementById(‘dungeon-btn’).addEventListener(‘click’, () => changeScreen(‘dungeon-select’));

Document.getElementById(‘inventory-btn’).addEventListener(‘click’, () => changeScreen(‘inventory’));

Document.getElementById(‘skill-btn’).addEventListener(‘click’, () => changeScreen(‘skill’));

Document.getElementById(‘upgrade-btn’).addEventListener(‘click’, () => changeScreen(‘upgrade’));

Document.getElementById(‘pets-btn’).addEventListener(‘click’, () => changeScreen(‘pets’));

Document.getElementById(‘mounts-btn’).addEventListener(‘click’, () => changeScreen(‘mounts’));

Document.getElementById(‘store-btn’).addEventListener(‘click’, () => changeScreen(‘store’));

Document.getElementById(‘altar-btn’).addEventListener(‘click’, () => {

If (gameData.character.level < 10) { addLog(‘The Mystic Altar is unlocked at Level 10.’); }

Else { changeScreen(‘altar’); }

});

Document.getElementById(‘shrine-btn’).addEventListener(‘click’, () => {

If (gameData.character.level < 5) { addLog(‘The Shrine is unlocked at Level 5.’); }

Else { changeScreen(‘shrine’); }

});

Document.getElementById(‘quest-btn’).addEventListener(‘click’, () => changeScreen(‘quest’));

Document.getElementById(‘guild-btn’).addEventListener(‘click’, () => {

If (gameData.character.level < 10) { addLog(‘The Guild Hall is unlocked at Level 10.’); }

Else { changeScreen(‘guild’); }

});

Document.getElementById(‘potion-btn’).addEventListener(‘click’, () => {

If (gameData.character.level < 5) { addLog(‘Potion Crafting is unlocked at Level 5.’); }

Else { changeScreen(‘potion’); }

});

Document.getElementById(‘unlocks-btn’).addEventListener(‘click’, () => changeScreen(‘unlocks’));

Document.querySelectorAll(‘.back-btn’).forEach(btn => btn.addEventListener(‘click’, () => {

If (isAutoCrawling && activeScreen !== ‘combat’) {

addLog(‘Auto-crawl is active, you can navigate freely.’);

changeScreen(‘main’);

} else {

changeScreen(‘main’);

}

}));

Document.getElementById(‘start-dungeon-btn’).addEventListener(‘click’, window.startDungeon);

Document.getElementById(‘auto-crawl-btn’).addEventListener(‘click’, window.toggleAutoCrawl);

</script>

</head>

<body class=”p-4 flex items-center justify-center”>

<div class=”container flex flex-col items-center”>

<div id=”select-class-screen” class=”w-full max-w-sm main-card rounded-xl p-6 flex flex-col items-center shadow-lg”>

<h1 class=”pixel-font text-2xl mb-6 text-center text-white”>Choose Your Class</h1>

<div id=”class-select-container” class=”space-y-4 w-full”>

<button onclick=”selectClass(‘thief’)” class=”btn w-full py-4 text-lg”>Thief</button>

<button onclick=”selectClass(‘sorcerer’)” class=”btn w-full py-4 text-lg”>Sorcerer</button>

<button onclick=”selectClass(‘archer’)” class=”btn w-full py-4 text-lg”>Archer</button>

</div>

</div>

<div id=”game-screen” class=”hidden w-full”>

<div id=”user-id-display” class=”text-sm text-gray-400 text-center mb-4 truncate”></div>

<div id=”main-screen”>

<div class=”main-card rounded-2xl p-6 flex flex-col space-y-6”>

<div class=”flex items-center space-x-4”>

<div class=”flex-shrink-0 w-20 h-20 rounded-full bg-gray-600 flex items-center justify-center text-3xl”>

<span id=”player-icon”></span>

</div>

<div class=”flex-grow”>

<h2 id=”character-name” class=”text-2xl font-bold text-white”></h2>

<ul id=”character-stats” class=”text-gray-300 text-sm mt-2 space-y-1”></ul>

</div>

</div>

<div class=”flex justify-between items-center bg-gray-800 p-3 rounded-xl shadow-inner”>

<div class=”flex items-center space-x-2”><i class=”fas fa-coins text-yellow-400”></i><span id=”gold” class=”text-white font-bold text-lg”></span></div>

<div class=”flex items-center space-x-2”><i class=”fas fa-gem text-purple-400”></i><span id=”gems” class=”text-white font-bold text-lg”></span></div>

<div class=”flex items-center space-x-2”><i class=”fas fa-flask text-teal-400”></i><span id=”essence” class=”text-white font-bold text-lg”></span></div>

</div>

<div>

<div class=”flex justify-between text-sm text-gray-400 mb-1”><span id=”level-info”></span></div>

<div class=”w-full bg-gray-700 rounded-full h-2.5”><div id=”xp-bar-inner” class=”bg-blue-500 h-2.5 rounded-full transition-all duration-300” style=”width: 0%;”></div></div>

</div>

<div>

<h3 class=”text-xl font-bold mb-2 text-white”>Equipment</h3>

<ul id=”equipment” class=”space-y-2 text-gray-300”></ul>

</div>

<div class=”grid grid-cols-2 gap-4”>

<button id=”dungeon-btn” class=”btn px-4 py-3”><i class=”fas fa-dungeon mr-2”></i>Dungeons</button>

<button id=”inventory-btn” class=”btn px-4 py-3”><i class=”fas fa-box-open mr-2”></i>Inventory</button>

<button id=”skill-btn” class=”btn px-4 py-3”><i class=”fas fa-tree mr-2”></i>Skills</button>

<button id=”upgrade-btn” class=”btn px-4 py-3”><i class=”fas fa-arrow-up mr-2”></i>Upgrade</button>

<button id=”pets-btn” class=”btn px-4 py-3”><i class=”fas fa-paw mr-2”></i>Pets</button>

<button id=”mounts-btn” class=”btn px-4 py-3”><i class=”fas fa-horse mr-2”></i>Mounts</button>

<button id=”store-btn” class=”btn px-4 py-3”><i class=”fas fa-store mr-2”></i>Store</button>

<button id=”shrine-btn” class=”btn px-4 py-3”><i class=”fas fa-scroll mr-2”></i>Shrine</button>

<button id=”altar-btn” class=”btn px-4 py-3”><i class=”fas fa-hand-sparkles mr-2”></i>Altar</button>

<button id=”quest-btn” class=”btn px-4 py-3”><i class=”fas fa-list-check mr-2”></i>Quests</button>

<button id=”guild-btn” class=”btn px-4 py-3”><i class=”fas fa-users mr-2”></i>Guild Hall</button>

<button id=”potion-btn” class=”btn px-4 py-3”><i class=”fas fa-flask mr-2”></i>Potions</button>

<button id=”unlocks-btn” class=”btn px-4 py-3”><i class=”fas fa-lock-open mr-2”></i>Unlocks</button>

</div>

</div>

</div>

<div id=”dungeon-select-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Choose a Dungeon</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<h4 class=”text-lg font-bold mt-4 mb-2”>Dungeon Type</h4>

<div class=”space-y-2”>

<div class=”flex items-center space-x-2”><input type=”radio” id=”normal” name=”dungeon-type” value=”Normal” checked><label for=”normal”>Normal (Standard)</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”gold” name=”dungeon-type” value=”Gold”><label for=”gold”>Gold (Extra Gold)</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”gem” name=”dungeon-type” value=”Gem”><label for=”gem”>Gem (Extra Gems)</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”chest” name=”dungeon-type” value=”Chest”><label for=”chest”>Chest (Chest Drops)</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”dimensional” name=”dungeon-type” value=”Dimensional”><label for=”dimensional”>Dimensional Rift (Infinite)</label></div>

</div>

<h4 class=”text-lg font-bold mt-4 mb-2”>Difficulty</h4>

<div class=”space-y-2”>

<div class=”flex items-center space-x-2”><input type=”radio” id=”easy” name=”dungeon-difficulty” value=”Easy” checked><label for=”easy”>Easy</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”medium” name=”dungeon-difficulty” value=”Medium”><label for=”medium”>Medium</label></div>

<div class=”flex items-center space-x-2”><input type=”radio” id=”hard” name=”dungeon-difficulty” value=”Hard”><label for=”hard”>Hard</label></div>

</div>

<div class=”grid grid-cols-2 gap-4 mt-6”>

<button id=”start-dungeon-btn” class=”btn w-full”>Start Crawl</button>

<button id=”auto-crawl-btn” class=”btn w-full bg-green-500”>Start Auto Crawl</button>

</div>

</div>

<div id=”combat-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

<div class=”flex flex-col items-center”>

<div class=”text-center mb-4”>

<span class=”text-4xl” id=”monster-icon”></span>

<h3 class=”text-2xl font-bold” id=”monster-name”></h3>

<div class=”w-full bg-red-800 rounded-full h-4 mt-2”>

<div id=”monster-hp-bar” class=”hp-bar-inner h-4 rounded-full” style=”width:100%;”></div>

</div>

<span id=”monster-hp-text” class=”text-sm mt-1”></span>

</div>

<div class=”text-center my-6”>

<span class=”text-4xl” id=”player-icon-combat”></span>

<div class=”w-full bg-green-800 rounded-full h-4 mt-2”>

<div id=”player-hp-bar” class=”hp-bar-inner h-4 rounded-full” style=”width:100%;”></div>

</div>

<span id=”player-hp-text” class=”text-sm mt-1”></span>

</div>

</div>

<div id=”combat-log” class=”h-32 overflow-y-auto text-sm bg-gray-800 p-2 rounded-lg border border-gray-600”></div>

<div class=”grid grid-cols-2 gap-4 mt-4”>

<button onclick=”attack()” class=”btn px-4 py-3”><i class=”fas fa-hand-fist mr-2”></i>Attack</button>

<button id=”active-ability-btn” onclick=”useActiveAbility()” class=”btn px-4 py-3”></button>

<button id=”mount-ability-btn” onclick=”useMountSkill()” class=”btn px-4 py-3”></button>

<button id=”cloak-ability-btn” onclick=”useCloakAbility()” class=”btn px-4 py-3”></button>

<button onclick=”endCombat(false)” class=”btn px-4 py-3 bg-red-500 hover:bg-red-700”><i class=”fas fa-flag mr-2”></i>Flee</button>

</div>

</div>

<div id=”inventory-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Inventory</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div id=”inventory-list” class=”space-y-2”></div>

</div>

<div id=”skill-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Skills</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div id=”skill-list” class=”space-y-2”></div>

</div>

<div id=”upgrade-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Character Upgrades</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p id=”skill-points-display” class=”text-lg font-bold mb-4”></p>

<div id=”upgrade-buttons” class=”grid grid-cols-1 md:grid-cols-2 gap-4”></div>

</div>

<div id=”pets-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Pets</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div id=”pets-list” class=”space-y-2”></div>

</div>

<div id=”mounts-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Mounts</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div id=”mounts-list” class=”space-y-2”></div>

</div>

<div id=”store-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Store</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div class=”tabs flex justify-around mb-4”>

<button class=”tab-btn active w-1/4 py-2 text-center rounded-t-lg bg-gray-700 text-white”>Pets</button>

<button class=”tab-btn w-1/4 py-2 text-center rounded-t-lg text-gray-400”>Mounts</button>

<button class=”tab-btn w-1/4 py-2 text-center rounded-t-lg text-gray-400”>Cloaks</button>

<button class=”tab-btn w-1/4 py-2 text-center rounded-t-lg text-gray-400”>Premium</button>

</div>

<div id=”pet-store-items” class=”tab-content”></div>

<div id=”mount-store-items” class=”tab-content hidden”></div>

<div id=”cloak-store-items” class=”tab-content hidden”></div>

<div id=”premium-store-items” class=”tab-content hidden”></div>

</div>

<div id=”altar-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Mystic Altar</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p class=”text-gray-400 mb-4”>Combine 3 items of the same rarity into a random item of the next rarity.</p>

<div id=”altar-list” class=”space-y-2”></div>

</div>

<div id=”shrine-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Shrine of Power</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p class=”text-gray-400 mb-4”>Offer 50 Gold and 10 Essence to receive a random blessing.</p>

<div class=”flex justify-center mb-4”>

<button onclick=”useShrine()” class=”btn px-8 py-4”>Make an Offering</button>

</div>

<div id=”shrine-result” class=”hidden p-4 mt-4 bg-gray-800 rounded-lg”>

<h4 class=”text-xl font-bold text-center text-white mb-2”>You Received:</h4>

<ul id=”shrine-rewards-list” class=”list-disc list-inside space-y-1”></ul>

</div>

</div>

<div id=”quest-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Daily Quests</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<div id=”quest-list” class=”space-y-4”></div>

</div>

<div id=”guild-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Guild Hall (Leaderboards)</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p class=”text-gray-400 mb-4”>Top Dungeon Crawlers by highest floor reached.</p>

<ul id=”leaderboard-list” class=”space-y-2”></ul>

</div>

<div id=”potion-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Potion Crafting</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p id=”potion-info” class=”mb-4”></p>

<h4 class=”text-lg font-bold mb-2”>Craft Potions</h4>

<div id=”potion-crafting-buttons” class=”grid grid-cols-1 gap-4 mb-4”></div>

<h4 class=”text-lg font-bold mb-2”>Use Potions</h4>

<div id=”potion-use-buttons” class=”grid grid-cols-1 gap-4”></div>

</div>

<div id=”unlocks-screen” class=”hidden main-card rounded-2xl p-6 mt-4”>

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

<h3 class=”text-2xl font-bold text-white”>Unlocks</h3>

<button class=”back-btn btn px-4 py-2 text-sm”>Back</button>

</div>

<p class=”text-gray-400 mb-4”>Explore what’s coming next as you level up!</p>

<ul id=”unlock-list” class=”list-none space-y-2 text-white”></ul>

</div>

</div>

</div>

<script>

// Tab switching logic for the store modal

Document.addEventListener(‘DOMContentLoaded’, () => {

Const tabs = document.querySelectorAll(‘.tab-btn’);

Const contents = document.querySelectorAll(‘.tab-content’);

Tabs.forEach(tab => {

Tab.addEventListener(‘click’, () => {

Tabs.forEach(t => t.classList.remove(‘active’, ‘bg-gray-700’, ‘text-white’));

Tabs.forEach(t => t.classList.add(‘text-gray-400’));

Tab.classList.add(‘active’, ‘bg-gray-700’, ‘text-white’);

Tab.classList.remove(‘text-gray-400’);

Contents.forEach(content => content.classList.add(‘hidden’));

Const target = document.getElementById(tab.textContent.toLowerCase().trim().replace(‘ ‘, ‘-‘) + ‘-store-items’);

If (target) {

Target.classList.remove(‘hidden’);

}

});

});

});

</script>

</body>

</html>