using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Linq;

using System.Threading;

class Program

{

private static SQLiteConnection connection;

static void Main()

{

InitializeDatabase();

bool showMenu = true;

while (showMenu)

{

Console.Clear();

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("Welcome to Character Creation!");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("1. Create a character");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("2. Delete a character");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("3. Display all characters");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("4. Exit");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.Write("Select: ");

Console.ForegroundColor = ConsoleColor.White;

string option = Console.ReadLine();

Console.Clear();

switch (option)

{

case "1":

Character character = Character.CreateCharacter();

break;

case "2":

Console.Clear();

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("Delete a Character");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

List<string> characterNames = Character.GetCharacterNames(connection);

Character.DisplayAllCharacters();

if (characterNames.Count == 0)

{

break;

}

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Press any key to return to the menu...");

Console.ForegroundColor = ConsoleColor.White;

ConsoleKeyInfo key = Console.ReadKey();

if (key.Key != ConsoleKey.Enter)

{

break;

}

Console.Clear();

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Enter the name of the character to delete or type 'M' to return to the menu: ");

Console.ForegroundColor = ConsoleColor.White;

string characterToDelete = Console.ReadLine().Trim();

if (characterToDelete.ToUpper() == "M")

{

break;

}

if (!string.IsNullOrWhiteSpace(characterToDelete))

{

Character.DeleteCharacterByName(characterToDelete, connection);

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Error: Please enter a valid character name or 'M' to return to the menu.");

Console.ForegroundColor = ConsoleColor.White;

}

break;

case "3":

Character.DisplayAllCharacters();

break;

case "4":

bool exitConfirmed = ConfirmExit();

if (exitConfirmed)

{

showMenu = false;

}

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Invalid option. Please select 1, 2, 3, or 4.\n");

Console.ForegroundColor = ConsoleColor.White;

break;

}

}

connection.Close();

}

public static void TextEffect(string text)

{

foreach (char c in text)

{

Console.Write(c);

Thread.Sleep(1);

}

Console.WriteLine();

}

private static void DisplayLoadingBar(int totalDots)

{

for (int i = 0; i < totalDots; i++)

{

Console.Write("█");

Thread.Sleep(20);

}

Console.WriteLine();

}

static bool ConfirmExit()

{

while (true)

{

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("Are you sure you want to exit?");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Type 'yes' to confirm, or 'no' to continue");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.Write("Your choice: ");

Console.ForegroundColor = ConsoleColor.White;

string userInput = Console.ReadLine().ToLower();

if (userInput == "yes")

{

Console.Clear();

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.WriteLine("║ Thank you and goodbye!");

Console.WriteLine("╚════════════════════════════════════════════════════");

return true;

}

else if (userInput == "no")

{

return false;

}

else

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[ERROR] Please enter 'yes' or 'no'.\n");

Console.ForegroundColor = ConsoleColor.White;

}

}

}

private static void InitializeDatabase()

{

string databasePath = "C:\\Users\\Thalia\\source\\repos\\Sample\\Sample\\char.db";

connection = new SQLiteConnection($"Data Source={databasePath};Version=3;");

connection.Open();

using (var command = new SQLiteCommand(

"CREATE TABLE IF NOT EXISTS Character (" +

"Id INTEGER PRIMARY KEY AUTOINCREMENT, " +

"Name TEXT, " +

"Gender TEXT, " +

"Race TEXT, " +

"SkinColor TEXT, " +

"FaceShape TEXT, " +

"HairType TEXT, " +

"HairColor TEXT, " +

"EyebrowType TEXT, " +

"EyebrowColor TEXT, " +

"EyeColor TEXT, " +

"NoseType TEXT, " +

"FacialHairType TEXT, " +

"AllyOfLight BOOLEAN, " +

"Dex INTEGER, " +

"Strength INTEGER, " +

"Agility INTEGER, " +

"Intelligence INTEGER);", connection))

{

command.ExecuteNonQuery();

}

}

class Character

{

private string name;

private string gender;

private string race;

private string skinColor;

private string faceShape;

private string hairType;

private string hairColor;

private string eyebrowType;

private string eyebrowColor;

private string eyeColor;

private string noseType;

private string facialHairType;

private bool allyOfLight;

private int dex;

private int strength;

private int agility;

private int intelligence;

private int Id { get; set; }

public static void DeleteCharacterByName(string name, SQLiteConnection connection)

{

using (var transaction = connection.BeginTransaction())

{

try

{

bool characterExists;

using (var checkCommand = new SQLiteCommand("SELECT COUNT(\*) FROM Character WHERE Name = @Name;", connection, transaction))

{

checkCommand.Parameters.AddWithValue("@Name", name);

characterExists = (long)checkCommand.ExecuteScalar() > 0;

}

if (characterExists)

{

TextEffect($"Deleting character '{name}'... ");

DisplayLoadingBar(40);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Character '{name}' successfully deleted. Press any key.");

Console.ForegroundColor = ConsoleColor.White;

Console.ReadKey();

using (var deleteCommand = new SQLiteCommand("DELETE FROM Character WHERE Name = @Name;", connection, transaction))

{

deleteCommand.Parameters.AddWithValue("@Name", name);

int rowsAffected = deleteCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

transaction.Commit();

Console.Clear();

DisplayAllCharacters();

}

else

{

transaction.Rollback();

Console.WriteLine($"Error: Character '{name}' was not deleted.");

}

}

}

else

{

Console.WriteLine($"Character '{name}' does not exist or is invalid.");

transaction.Rollback();

}

}

catch (Exception ex)

{

transaction.Rollback();

Console.WriteLine($"Error: {ex.Message}");

}

}

}

public static Character CreateCharacter()

{

Character character = new Character();

character.GetCharacterInfo();

character.GetDispositionAlignment();

character.GetStatPointAllocation();

SaveCharacterToDatabase(character);

return character;

}

public static void DisplayAllCharacters()

{

using (var command = new SQLiteCommand("SELECT \* FROM Character;", connection))

{

using (var reader = command.ExecuteReader())

{

if (!reader.HasRows)

{

Console.Clear();

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("No characters to display.");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.WriteLine("Press any key...");

Console.ReadKey();

return;

}

do

{

while (reader.Read())

{

Character character = MapRowToCharacter(reader);

character.DisplayCharacter();

}

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Do you want to delete a character (D) or go back to the menu (M)? ");

Console.ForegroundColor = ConsoleColor.White;

string userInput = Console.ReadLine().Trim().ToUpper();

if (userInput == "D")

{

TextEffect("Enter the name of the character to delete... ");

string characterNameToDelete = Console.ReadLine().Trim();

DeleteCharacterByName(characterNameToDelete, connection);

}

else if (userInput == "M")

{

break;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[ERROR]. Please enter 'D' to delete a character or 'M' to go back to the menu.");

Console.ForegroundColor = ConsoleColor.White;

}

} while (true);

}

}

}

public static void DisplayCharacterNames(SQLiteConnection connection)

{

List<string> characterNames = GetAllCharacterNames(connection);

if (characterNames.Count == 0)

{

Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("No characters found in the database. Delete option not available.");

Console.ForegroundColor = ConsoleColor.White;

return;

}

foreach (string name in characterNames)

{

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect($"{name}");

Console.ForegroundColor = ConsoleColor.White;

}

}

public static List<string> GetAllCharacterNames(SQLiteConnection connection)

{

List<string> characterNames = new List<string>();

using (var command = new SQLiteCommand("SELECT Name FROM Character;", connection))

{

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

characterNames.Add(reader["Name"].ToString());

}

}

}

return characterNames;

}

private static Character MapRowToCharacter(SQLiteDataReader reader)

{

Character character = new Character();

character.name = reader["Name"].ToString();

character.gender = reader["Gender"].ToString();

character.race = reader["Race"].ToString();

character.skinColor = reader["SkinColor"].ToString();

character.faceShape = reader["FaceShape"].ToString();

character.hairType = reader["HairType"].ToString();

character.hairColor = reader["HairColor"].ToString();

character.eyebrowType = reader["EyebrowType"].ToString();

character.eyebrowColor = reader["EyebrowColor"].ToString();

character.eyeColor = reader["EyeColor"].ToString();

character.noseType = reader["NoseType"].ToString();

character.facialHairType = reader["FacialHairType"].ToString();

character.allyOfLight = Convert.ToBoolean(reader["AllyOfLight"]);

character.dex = Convert.ToInt32(reader["Dex"]);

character.strength = Convert.ToInt32(reader["Strength"]);

character.agility = Convert.ToInt32(reader["Agility"]);

character.intelligence = Convert.ToInt32(reader["Intelligence"]);

return character;

}

private static void CharacterExists(string name)

{

using (var command = new SQLiteCommand("SELECT COUNT(\*) FROM Character WHERE Name = @Name;", connection))

{

command.Parameters.AddWithValue("@Name", name);

if ((long)command.ExecuteScalar() > 0) { }

}

}

private static void SaveCharacterToDatabase(Character character)

{

try

{

CharacterExists(character.name);

Console.ForegroundColor = ConsoleColor.Yellow;

TextEffect("Saving character to the database... ");

Console.ForegroundColor = ConsoleColor.White;

DisplayLoadingBar(40);

using (var command = new SQLiteCommand(

"INSERT INTO Character (Name, Gender, Race, SkinColor, FaceShape, HairType, HairColor, " +

"EyebrowType, EyebrowColor, EyeColor, NoseType, FacialHairType, AllyOfLight, " +

"Dex, Strength, Agility, Intelligence) " +

"VALUES (@Name, @Gender, @Race, @SkinColor, @FaceShape, @HairType, @HairColor, " +

"@EyebrowType, @EyebrowColor, @EyeColor, @NoseType, @FacialHairType, @AllyOfLight, " +

"@Dex, @Strength, @Agility, @Intelligence);", connection))

{

command.Parameters.AddWithValue("@Name", character.name);

command.Parameters.AddWithValue("@Gender", character.gender);

command.Parameters.AddWithValue("@Race", character.race);

command.Parameters.AddWithValue("@SkinColor", character.skinColor);

command.Parameters.AddWithValue("@FaceShape", character.faceShape);

command.Parameters.AddWithValue("@HairType", character.hairType);

command.Parameters.AddWithValue("@HairColor", character.hairColor);

command.Parameters.AddWithValue("@EyebrowType", character.eyebrowType);

command.Parameters.AddWithValue("@EyebrowColor", character.eyebrowColor);

command.Parameters.AddWithValue("@EyeColor", character.eyeColor);

command.Parameters.AddWithValue("@NoseType", character.noseType);

command.Parameters.AddWithValue("@FacialHairType", character.facialHairType);

command.Parameters.AddWithValue("@AllyOfLight", character.allyOfLight);

command.Parameters.AddWithValue("@Dex", character.dex);

command.Parameters.AddWithValue("@Strength", character.strength);

command.Parameters.AddWithValue("@Agility", character.agility);

command.Parameters.AddWithValue("@Intelligence", character.intelligence);

command.ExecuteNonQuery();

}

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Character successfully saved to the database.");

Console.ForegroundColor = ConsoleColor.White;

}

catch (InvalidOperationException ex)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Error: {ex.Message}");

Console.ForegroundColor = ConsoleColor.White;

}

catch (Exception ex)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Error: {ex.Message}");

Console.ForegroundColor = ConsoleColor.White;

}

}

public void DisplayCharacter()

{

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("Character Display:");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Name: {name}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Gender: {gender}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Race: {race}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Skin Color: {skinColor}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Face Shape: {faceShape}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Hair Type: {hairType}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Hair Color: {hairColor}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Eyebrow Type: {eyebrowType}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Eyebrow Color: {eyebrowColor}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Eye Color: {eyeColor}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Nose Type: {noseType}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Facial Hair Type: {facialHairType}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Ally of Light: {allyOfLight}");

Console.ForegroundColor = ConsoleColor.White;

if (dex + strength + agility + intelligence > 0)

{

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Dexterity: {dex}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Strength: {strength}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Agility: {agility}");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"Intelligence: {intelligence}");

Console.ForegroundColor = ConsoleColor.White;

}

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

}

private void GetCharacterInfo()

{

while (true)

{

name = GetValidatedInput("Character Name");

if (String.IsNullOrWhiteSpace(name))

{

TextEffect("OBOB bawal walang laman!!!!!!!!!!");

Console.Clear();

}

else if (IsCharacterNameUnique(name))

{

break;

}

else

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"[ERROR] '{name}' already exists. Please choose a different name.");

Console.ForegroundColor = ConsoleColor.White;

}

}

Console.Clear();

gender = GetValidatedInput("Character Gender", new[] { "Male", "Female" });

Console.Clear();

race = GetValidatedInput("Character Race", new[] { "Orc", "Human", "Elf", "Dwarf", "Undead" });

Console.Clear();

skinColor = GetValidatedInput("Character Skin Color", new[] { "Red", "Green", "Blue", "Black", "White" });

Console.Clear();

faceShape = GetValidatedInput("Character Face Shape", new[] { "Round", "Diamond", "Heart", "Oblong", "Square" });

Console.Clear();

hairType = GetValidatedInput("Character Hair Type", new[] { "Bald", "Dreadlocks", "Curly", "Straight", "Coily" });

Console.Clear();

hairColor = GetValidatedInput("Character Hair Color", new[] { "Red", "Green", "Blue", "Black", "White" });

Console.Clear();

eyebrowType = GetValidatedInput("Character Eyebrow Type", new[] { "Straight", "Rounded", "S-Shape", "Steep Arch", "Arched" });

Console.Clear();

eyebrowColor = GetValidatedInput("Character Eyebrow Color", new[] { "Red", "Green", "Blue", "Black", "White" });

Console.Clear();

eyeColor = GetValidatedInput("Character Eye Color", new[] { "Red", "Green", "Blue", "Black", "White" });

Console.Clear();

noseType = GetValidatedInput("Character Nose Type", new[] { "Flat", "Big", "Pointed", "Hooked", "Small" });

Console.Clear();

facialHairType = GetValidatedInput("Character Facial Hair Type", new[] { "Subtle", "Mustache", "Beard", "Thick", "Goatee" });

Console.Clear();

}

private bool IsCharacterNameUnique(string name)

{

using (var command = new SQLiteCommand("SELECT COUNT(\*) FROM Character WHERE Name = @Name;", connection))

{

command.Parameters.AddWithValue("@Name", name);

return (long)command.ExecuteScalar() == 0;

}

}

private void GetDispositionAlignment()

{

allyOfLight = GetBooleanInput("Are you an Ally of Light?");

}

private void GetStatPointAllocation()

{

int totalStatPoints = 40;

Console.Clear();

Console.WriteLine("Available Stats:");

Console.WriteLine("1. Dexterity");

Console.WriteLine("2. Strength");

Console.WriteLine("3. Agility");

Console.WriteLine("4. Intelligence");

Console.WriteLine();

dex = GetStatInput("Dexterity", totalStatPoints);

totalStatPoints -= dex;

strength = GetStatInput("Strength", totalStatPoints);

totalStatPoints -= strength;

agility = GetStatInput("Agility", totalStatPoints);

totalStatPoints -= agility;

intelligence = GetStatInput("Intelligence", totalStatPoints);

}

private string GetValidatedInput(string prompt, string[] validOptions = null)

{

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"{prompt}");

if (validOptions != null)

{

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

for (int i = 0; i < validOptions.Length; i++)

{

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect($"{i + 1}: {validOptions[i]}");

Console.ForegroundColor = ConsoleColor.White;

}

}

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Your choice: ");

Console.ForegroundColor = ConsoleColor.White;

string userInput = Console.ReadLine();

if (validOptions == null || (int.TryParse(userInput, out int choice) && choice >= 1 && choice <= validOptions.Length))

{

return validOptions != null ? validOptions[int.Parse(userInput) - 1] : userInput;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[ERROR]. Please choose a valid option.");

Console.ForegroundColor = ConsoleColor.White;

return GetValidatedInput(prompt, validOptions);

}

}

private bool GetBooleanInput(string prompt)

{

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"{prompt}");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("1: true");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("2: false");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Your choice: ");

Console.ForegroundColor = ConsoleColor.White;

string userInput = Console.ReadLine();

Console.Clear();

if (userInput == "1")

{

return true;

}

else if (userInput == "2")

{

return false;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[ERROR] Please enter '1' for true or '2' for false.");

Console.ForegroundColor = ConsoleColor.White;

return GetBooleanInput(prompt);

}

}

public static List<string> GetCharacterNames(SQLiteConnection connection)

{

List<string> characterNames = new List<string>();

using (var command = new SQLiteCommand("SELECT Name FROM Character;", connection))

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

characterNames.Add(reader["Name"].ToString());

}

}

return characterNames;

}

private int GetStatInput(string statName, int maxTotalPoints)

{

int remainingPoints = maxTotalPoints;

int stat = 0;

while (remainingPoints > 0)

{

Console.Clear();

Console.WriteLine("╔════════════════════════════════════════════════════");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("Available stats");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Dexterity");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Strength");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Agility");

Console.ForegroundColor = ConsoleColor.White;

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect("Intelligence");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╟────────────────────────────────────────────────────");

Console.Write("║ ");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect($"Remaining points: {remainingPoints}");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("╚════════════════════════════════════════════════════");

Console.ForegroundColor = ConsoleColor.Cyan;

TextEffect($"Allocate points for {statName}: ");

Console.ForegroundColor = ConsoleColor.White;

string userInput = Console.ReadLine();

try

{

stat = int.Parse(userInput);

if (stat < 0 || stat > Math.Min(maxTotalPoints, remainingPoints))

{

throw new ArgumentOutOfRangeException();

}

remainingPoints -= stat;

if (remainingPoints > 0)

{

break;

}

}

catch (FormatException)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("[ERROR] Please enter a valid number.\n");

Console.ForegroundColor = ConsoleColor.White;

}

catch (ArgumentOutOfRangeException)

{

Console.Clear();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"[ERROR] Please enter a number between 0 and {Math.Min(maxTotalPoints, remainingPoints)}.\n");

Console.ForegroundColor = ConsoleColor.White;

}

}

return stat;

}

}

}