Gamesino Technical Manual

Final 4th Year Project

South East Technological University

Student: Samuel David

Student ID: C0025010

Supervisor: Doyle, Greg

Submission Date: 17/04/23

Table Of Contents

Introduction

The aim of this Techinal Manual is to show the code in its entirety for the project. The code will be labelled for easy viewing and each code snippet will be provided for each controller, model View etc.

Github is here : https://github.com/SamuelDavid60322/GamesPlatform

Components

ShoppingCartSummary

```
public class ShoppingCartSummary : ViewComponent
{
    private readonly ShoppingCart _shoppingCart;
    0 references
    public ShoppingCartSummary(ShoppingCart shoppingCart)
    {
        _shoppingCart = shoppingCart;

    }
    0 references
    public IViewComponentResult Invoke()
    {
        var items = _shoppingCart.GetShoppingCartItems();
        _shoppingCart.ShoppingCartItems = items;

        var shoppingCartViewModel = new ShoppingCartViewModel
        {
            ShoppingCart = _shoppingCart,
            ShoppingCartTotal = _shoppingCart.GetShoppingCartTotal()
        };
        return View(shoppingCartViewModel);
    }
}
```

WalletBalance

```
using GamesPlatform.Interfaces;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System.Runtime.ConstrainedExecution;
using System.Security.Claims;

namespace GamesPlatform.Components
{

    public class WalletBalance : ViewComponent
    {
        private readonly IWalletRepository _walletRepository;
        private readonly IHttpContextAccessor _httpContextAccessor;

        public WalletBalance(IWalletRepository walletRepository, IHttpContextAccessor httpContextAccessor)
        {
            _walletRepository = walletRepository;
            _httpContextAccessor = httpContextAccessor;
        }

        public IViewComponentResult Invoke()
        {
            string userId = _httpContextAccessor.HttpContext.User.FindFirstValue(ClaimTypes.NameIdentifier);
            decimal balance = _walletRepository.GetBalance(userId);
            var walletViewModel = new WalletViewModel
            {
                Balance = balance,
            };
            return View(walletViewModel);
        }
    }
}
```

Controllers

AdminController

```
using GamesPlatform.Data;

using GamesPlatform.Interfaces;

using GamesPlatform.Models;

using GamesPlatform.ViewModels;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Drawing.Text;



namespace GamesPlatform.Controllers

{

    [Authorize(Policy = "AdminOnly")]

    public class AdminController : Controller

    {

        private readonly RoleManager<IdentityRole> _roleManager;

        private readonly ApplicationDbContext _applicationDbContext;

        private readonly IGamesRepository _gamesRepository;

        private readonly IVerificationRepository _verificationRepository;
```

```
        public AdminController(RoleManager<IdentityRole> roleManager,
ApplicationDbContext applicationDbContext, IGamesRepository
gamesRepository, IVerificationRepository verificationRepository)

        {

            _roleManager = roleManager;

            _applicationDbContext = applicationDbContext;

            _gamesRepository = gamesRepository;

            _verificationRepository = verificationRepository;

        }

        public IActionResult Index()

        {

            return View();

        }



        [HttpGet]

        public IActionResult CreateRole()

        {

            return View();

        }

        [HttpPost]

        public async Task<IActionResult> CreateRole(CreateRoleViewModel
model)

        {

            if(ModelState.IsValid)
```

```
            {
                IdentityRole identityRole = new IdentityRole
                {
                    Name= model.RoleName
                };

                IdentityResult result = await
_roleManager.CreateAsync(identityRole);

                if(result.Succeeded)
                {
                    return RedirectToAction("Index", "Home");
                }
                foreach(IdentityError error in result.Errors)
                {
                    ModelState.AddModelError("", error.Description);
                }
            }


            return View(model);
        }
```

```
        public async Task<IActionResult> ViewVerificationRequests(string
sortOrder, string currentFilter, string searchString, int? pageNumber)

        {

            ViewData["CurrentSort"] = sortOrder;

            ViewData["StatusSortParam"] = String.IsNullOrEmpty(sortOrder)
? "status_desc" : "";

            ViewData["DateOfRequestSortParam"] = sortOrder ==
"DateOfRequest" ? "dateofrequest_desc" : "DateOfRequest";

            if (searchString != null)

            {

                pageNumber = 1;

            }

            else

            {

                searchString = currentFilter;

            }



            ViewData["CurrentFilter"] = searchString;

            var verifications = from v in
_applicationDbContext.Verifications select v;

            if (!String.IsNullOrEmpty(searchString))

            {

                verifications = verifications.Where(v =>
v.UserID.Contains(searchString));

            }
```

```
            switch (sortOrder)
            {
                case "status_desc":
                    verifications = verifications.OrderByDescending(v =>
v.Status);
                    break;
                case "DateOfRequest":
                    verifications = verifications.OrderBy(v =>
v.DateOfRequest);
                    break;
                case "dateofrequest_desc":
                    verifications = verifications.OrderByDescending(v =>
v.DateOfRequest);
                    break;
                default:
                    verifications = verifications.OrderBy(v =>
v.DateOfRequest);
                    break;
            }
            int pageSize = 10;
            return View(await
PaginatedList<Verification>.CreateAsync(verifications.AsNoTracking(),
pageNumber ?? 1, pageSize));
        }
        public IActionResult UpdateVerificationRequest(int?
verificationId)
```

```
        {
            if (verificationId == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error", new { code
= 404 });
            }
            var verificationRequest =
_verificationRepository.RetrieveVerificationByID((int)verificationId);
            if (verificationRequest == null)
            {
                Response.StatusCode = 404;
                return RedirectToAction("HandleError", "Error", new { code
= 404 });
            }
            using (MemoryStream memoryStream = new
MemoryStream(verificationRequest.Content))
            {
                ViewData["Image"] =
Convert.ToBase64String(verificationRequest.Content);
            }

            return View(verificationRequest);
        }
```

```
        [HttpPost, ActionName("UpdateVerificationRequest")]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult>
UpdateVerificationRequestPost(int? verificationId)

        {

            if (verificationId == null)

            {

                Response.StatusCode = 404;

                return RedirectToAction("HandleError", "Error", new { code
= 404 });

            }

            var verificationRequestToUpdate = await
_applicationDbContext.Verifications.FirstOrDefaultAsync(g =>
g.VerificationID == verificationId);

            if (await TryUpdateModelAsync<Verification>(

                verificationRequestToUpdate,

                "",

                v => v.Status))

            {

                try

                {

                    await _applicationDbContext.SaveChangesAsync();

                    return RedirectToAction("ViewVerificationRequests");

                }

                catch (DbUpdateException)
```

```
                {

                    ModelState.AddModelError("", "Unable to save changes
to the database");

                }

            }

            return View(verificationRequestToUpdate);

        }

    }

}
```

BlackJackController

```
using GamesPlatform.Data;

using GamesPlatform.Interfaces;

using GamesPlatform.Models;

using GamesPlatform.ViewModels;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Security.Claims;



namespace GamesPlatform.Controllers

{
```

```
    public class BlackJackController : Controller

    {

        private readonly ApplicationDbContext _applicationDbContext;

        private readonly IWalletRepository _walletRepository;



        public BlackJackController(ApplicationDbContext
applicationDbContext, IWalletRepository walletRepository)

        {

            _applicationDbContext = applicationDbContext;

            _walletRepository = walletRepository;

        }

        public IActionResult Index()

        {

            return View();

        }



        [HttpPost]

        public async Task<IActionResult> Play(BlackJackViewModel model)

        {

            model.PlayerHand = DrawCards(2);
```

```
            model.DealerHand = DrawCards(2);



            int playerScore = CalculateHandScore(model.PlayerHand);

            int dealerScore = CalculateHandScore(model.DealerHand);



            if (playerScore > 21)

            {

                model.IsWinner = false;

            }

            else if (dealerScore > 21 || playerScore > dealerScore)

            {

                model.IsWinner = true;

            }

            else

            {

                model.IsWinner = false;

            }



            model.AmountWon = model.IsWinner.Value ? 10 : 0; // Change
this to the desired win amount
```

```
            var casinoResult = new CasinoResult

            {

                UserChoice = $"Player: {string.Join(",",
model.PlayerHand)}; Dealer: {string.Join(",", model.DealerHand)}",

                ComputerChoice = $"Player: {string.Join(",",
model.PlayerHand)}; Dealer: {string.Join(",", model.DealerHand)}",

                Result = model.IsWinner.Value ? "Win" : "Lose",

                Win = model.IsWinner.Value,

                AmountWon = model.AmountWon,

                DateResultPlaced = DateTime.Now

            };



            _applicationDbContext.CasinoResults.Add(casinoResult);

            await _applicationDbContext.SaveChangesAsync();



            if (model.IsWinner.Value)

            {

                string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);

                _walletRepository.AddToWallet(userId, model.AmountWon);
```

```
        }

        model.CasinoResult = casinoResult;

        return View("Index", model);
    }

    private List<int> DrawCards(int count)
    {
        var cards = new List<int>();
        var rand = new Random();
        for (int i = 0; i < count; i++)
        {
            cards.Add(rand.Next(1, 11));
        }
        return cards;
    }

    private int CalculateHandScore(List<int> hand)
```

```
        {
            int score = 0;
            foreach (int card in hand)
            {
                score += card;
            }
            return score;
        }
    }
}
```

CasinoController

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Razor;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.Linq;
using System.Security.Claims;
```

```
namespace GamesPlatform.Controllers
{
    [Authorize]
    public class CasinoController : Controller
    {
        private readonly IGamesRepository _gamesRepository;
        private readonly ICasinoResultRepository _casinoResultRepository;
        private readonly IWalletRepository _walletRepository;
        private readonly IVerificationRepository _verificationRepository;

        public CasinoController(IGamesRepository gamesRepository,
ICasinoResultRepository casinoResultRepository, IWalletRepository
walletRepository, IVerificationRepository verificationRepository)
        {
            _gamesRepository = gamesRepository;
            _casinoResultRepository = casinoResultRepository;
            _walletRepository = walletRepository;
            _verificationRepository = verificationRepository;
        }
        [Authorize]
        public IActionResult Index()
        {
            if (User.Identity.IsAuthenticated)
```

```
            {

                string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;

                var verification =
_verificationRepository.RetrieveVerificationByUserID(userId);




                if (verification == null)

                {

                    return RedirectToAction("Index", "Verification");

                }

                else if (verification.Status == "Pending")

                {

                    ViewBag.VerificationMessage = "Your Verification
request to play casino Games is being reviewed by admins, Check back
later.";

                }

                else if (verification.Status == "Denied")

                {

                    ViewBag.VerificationMessage = "Sorry, but your request
to play casino games has been denied. Email support if you think this a
mistake.";

                }

            }
```

```
            var casinoViewModel = new CasinoViewModel

            {

                CasinoGame = _gamesRepository.CasinoGames

            };

            return View(casinoViewModel);

        }



    }

}
```

CoinFlipController

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class CoinFlipController : Controller
    {
        private readonly ApplicationDbContext _applicationDbContext;
        private readonly IWalletRepository _walletRepository;
        public CoinFlipController(ApplicationDbContext
applicationDbContext, IWalletRepository walletRepository)
        {
```

```
            _applicationDbContext = applicationDbContext;
            _walletRepository = walletRepository;
        }
        public IActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public async Task<IActionResult> Flip(CoinFlipViewModel model)
        {
            var rand = new Random();
            int outcome = rand.Next(2);
            string coinSide = outcome == 0 ? "Heads" : "Tails";

            model.Result = coinSide;
            model.IsWinner = coinSide.Equals(model.PlayerChoice,
StringComparison.OrdinalIgnoreCase);

            var casinoResult = new CasinoResult
            {
                UserChoice = model.PlayerChoice,
                ComputerChoice = coinSide,
                Result = model.IsWinner.Value ? "Win" : "Lose",
                Win = model.IsWinner.Value,
                AmountWon = model.IsWinner.Value ? 5 : 0, // Change this
to the desired win amount
                DateResultPlaced = DateTime.Now
            };

            _applicationDbContext.CasinoResults.Add(casinoResult);
            await _applicationDbContext.SaveChangesAsync();

            if (model.IsWinner.Value)
            {
                string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
                _walletRepository.AddToWallet(userId,
casinoResult.AmountWon);
            }
```

```
            model.CasinoResult = casinoResult;

            return View("Index", model);
        }
    }
}
```

ContactController

```
using Microsoft.AspNetCore.Mvc;

namespace GamesPlatform.Controllers
{
    public class ContactController : Controller
    {
        public ViewResult Index()
        {
            return View();
        }
    }
}
```

GamesController

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
```

```
    public class GamesController :Controller
    {
        private readonly ICategoryRepository _categoryRepository;
        private readonly IGamesRepository _gamesRepository;
        private readonly ApplicationDbContext _applicationDbContext;

        public GamesController(ICategoryRepository categoryRepository,
IGamesRepository gamesRepository, ApplicationDbContext
applicationDbContext)
        {
            _categoryRepository = categoryRepository;
            _gamesRepository = gamesRepository;
            _applicationDbContext = applicationDbContext;

        }

        public ViewResult List(string category, string searchString =
null)
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var purchasedGameIDs = new List<int>();
            if (userId != null)
            {
                purchasedGameIDs = _applicationDbContext.OrderDetails
                    .Include(od => od.Order)
                    .Where(od => od.Order.UserID == userId)
                    .Select(od => od.GameID)
                    .ToList();
            }

            IEnumerable<Game> games;
            string currentCategory;
            //if category is null then current category is all games
            if (string.IsNullOrEmpty(category))
            {
                games = _gamesRepository.Games.OrderBy(g => g.GameID);
                currentCategory = "All Games";
            }
            else
            {
```

```
                //find games that matches category name argument
                games = _gamesRepository.Games.Where(c =>
c.Category.CategoryName == category);
                //find category that matches current category, with a null
check that will display the category name
                currentCategory =
_categoryRepository.Categories.FirstOrDefault(currentCategory =>
currentCategory.CategoryName == category)?.CategoryName;
            }

            if (!string.IsNullOrEmpty(searchString))
            {
                games = games.Where(g => g.GameName.Contains(searchString,
StringComparison.OrdinalIgnoreCase));
            }

            ViewData["searchString"] = searchString;
            return View(new GameListViewModel
            {
                Games = games,
                CurrentCategory = currentCategory,
                PurchasedGameIDs = purchasedGameIDs
            });
        }


        public ViewResult Details(int gameId)
        {
            var game = _gamesRepository.GetGamesById(gameId);
            if (game == null)
            {
                return View("~/Views/Error/Error.cshtml");
            }
            string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
            bool userOwnsGame = UserOwnsGame(userId, gameId);
            ViewData["UserHasGame"] = userOwnsGame;
            return View(game);
        }
```

```
        public bool UserOwnsGame(string userId, int gameId)
        {
            // Assuming you have the ApplicationDbContext and the
OrderDetails DbSet
            var userOwnsGame =
_applicationDbContext.OrderDetails.Include(od => od.Order).Any(od =>
od.Order.UserID == userId && od.GameID == gameId);
            return userOwnsGame;
        }
        public IActionResult hol()
        {
            return View();
        }

    }
}
```

HomeController

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    public class HomeController : Controller
    {

        private readonly IGamesRepository _gamesRepository;
        private readonly ApplicationDbContext _applicationDbContext;
```

```
        public HomeController(IGamesRepository gamesRepository,
ApplicationDbContext applicationDbContext)
        {
            _gamesRepository = gamesRepository;
            _applicationDbContext = applicationDbContext;
        }

        public ViewResult Index()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            var purchasedGameIDs = new List<int>();
            if (userId != null)
            {
                purchasedGameIDs = _applicationDbContext.OrderDetails
                    .Include(od => od.Order)
                    .Where(od => od.Order.UserID == userId)
                    .Select(od => od.GameID)
                    .ToList();
            }

            var homeViewModel = new HomeViewModel
            {
                FeaturedGames = _gamesRepository.FeaturedGames,
                FreeGames = _gamesRepository.FreeGames,
                PurchasedGameIDs = purchasedGameIDs
            };
            return View(homeViewModel);
        }

        public IActionResult Privacy()
        {
            return View();
        }

        public IActionResult Libary()
        {
            return View();
        }

        public IActionResult Error404(int? statusCode = null)
```

```
        {
            if (statusCode.HasValue && statusCode.Value == 404)
            {
                return View();
            }

            return RedirectToAction("Index");
        }

        [ResponseCache(Duration = 0, Location =
ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

LibraryController

```
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class LibraryController : Controller
    {
        private readonly IGamesRepository _gamesRepository;
        private readonly IOrderRepository _orderRepository;
        private readonly ApplicationDbContext _applicationDbContext;
```

```
        public LibraryController(IOrderRepository orderRepository,
IGamesRepository gamesRepository, ApplicationDbContext
applicationDbContext)
        {
            _orderRepository = orderRepository;
            _gamesRepository = gamesRepository;
            _applicationDbContext = applicationDbContext;
        }
        [Authorize]
        public IActionResult Index()
        {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var orders = _applicationDbContext.Orders.Include(o =>
o.OrderDetails).ThenInclude(od => od.Game).Where(o => o.UserID==
userId).ToList();

                var purchasedGames = orders.SelectMany(o =>
o.OrderDetails).Select(od => od.Game).Distinct().ToList();

            var viewModel = new LibraryViewModel
            {
                PurchasedGames = purchasedGames
            };
            return View(viewModel);
        }
    }
}
```

OrderController

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
```

```
using System.Diagnostics.Metrics;
using System.Reflection.Emit;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class OrderController : Controller
    {
        private readonly IOrderRepository _orderRepository;
        private readonly ShoppingCart _shoppingCart;

        public OrderController(IOrderRepository orderRepository,
ShoppingCart shoppingCart)
        {
            _orderRepository = orderRepository;
            _shoppingCart = shoppingCart;

        }

        public IActionResult Checkout()
        {
            return View();
        }

        [HttpPost]
        [Authorize]
        public IActionResult Checkout(Order order)
        {
            var items = _shoppingCart.GetShoppingCartItems();
            _shoppingCart.ShoppingCartItems = items;

            if (_shoppingCart.ShoppingCartItems.Count ==0)
            {
                ModelState.AddModelError("", "Cart is Empty");
            }

            if(ModelState.IsValid)
            {
```

```
                string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
                string firstName = "";
                string lastName = "";
                string addressLine1 = "";
                string addressLine2 = "";
                string zipCode = "";
                string city = "";
                string country = "";
                string phoneNumber = "";
                string email =
User.FindFirstValue(ClaimTypes.NameIdentifier);
                int orderId = 0;
                order.OrderID = orderId;

                //var orderDetails =
_orderDetailRepository.GetAllOrderDetails(orderId);
                _orderRepository.CreateOrder(items, userId, firstName,
lastName, addressLine1, addressLine2, zipCode, city, country, phoneNumber,
email);

                _shoppingCart.ClearCart();
                return RedirectToAction("Index", "Payment");
            }
            return View(order);
        }

        public IActionResult CheckoutComplete()
        {
            var items = _shoppingCart.GetShoppingCartItems();
            return View("~/Views/Payment/Success");
        }

        public IActionResult Index()
        {
            string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);


            var orders = _orderRepository.GetOrdersByUserID(userId);
```

```
            return View(orders);

        }
    }
}
```

PaymentsController

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PayPal.Api;
using Stripe;
using Stripe.Checkout;
using System.ComponentModel;
using System.Diagnostics.Metrics;
using System.Reflection.Emit;
using System.Security.Claims;
using System.Globalization;
using static System.Net.WebRequestMethods;
using System.Net;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class PaymentsController : Controller
    {
        private readonly ShoppingCart _shoppingCart;
        private readonly IOrderRepository _orderRepository;
        private readonly ApplicationDbContext _applicationDbContext;
        private readonly IWalletRepository _walletRepository;
```

```
        public PaymentsController(IOrderRepository orderRepository,
ShoppingCart shoppingCart, ApplicationDbContext applicationDbContext,
IWalletRepository walletRepository)
        {
            _orderRepository = orderRepository;
            _shoppingCart = shoppingCart;
            _applicationDbContext = applicationDbContext;
            _walletRepository = walletRepository;

            StripeConfiguration.ApiKey =
"sk_test_51MLfkELqDZptVo03ItEDQzIaHydvHEVvyIw7SV0Z0GmzqsDWyxV3lEWLkHzfnr4n
rnxxInRobfod8LpmiLdlBqSw00oEP5ziP9";
        }
        [Authorize]

        [HttpPost("create-checkout-session")]
        public IActionResult CreateCheckoutSession()
        {
             decimal orderTotal;
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            orderTotal = _shoppingCart.GetShoppingCartTotal() * 100;

            var options = new SessionCreateOptions
            {
                LineItems = new List<SessionLineItemOptions>
        {
          new SessionLineItemOptions
          {
            PriceData = new SessionLineItemPriceDataOptions
            {
              UnitAmount = (long?)orderTotal,
              Currency = "eur",
              ProductData = new SessionLineItemPriceDataProductDataOptions
              {
                Name = "Total Amount",
              },
            },
            Quantity = 1,
          },
```

```
        },
                Mode = "payment",
                SuccessUrl =
"https://gamesino.azurewebsites.net/Payments/Success?session_id={CHECKOUT_
SESSION_ID}",
                CancelUrl =
"https://gamesino.azurewebsites.net/Payments/Cancel",
            };

            var service = new SessionService();
            Session session = service.Create(options);

            Response.Headers.Add("Location", session.Url);
            return new StatusCodeResult(303);
        }

        [Authorize]
        [HttpPost("create-funds-checkout-session")]
        public IActionResult CreateFundsCheckoutSession(decimal amount)
        {
            // Ensure the amount is valid before proceeding
            if (amount <= 0)
            {
                return BadRequest("Invalid amount.");
            }
            string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);

            var options = new SessionCreateOptions
            {
                LineItems = new List<SessionLineItemOptions>
        {
            new SessionLineItemOptions
            {
                PriceData = new SessionLineItemPriceDataOptions
                {
                    UnitAmount = (long)(amount * 100),
                    Currency = "eur",
                    ProductData = new
SessionLineItemPriceDataProductDataOptions
```

```
                    {
                        Name = $"Add {amount}€ to Wallet",
                    },
                },
                Quantity = 1,
            },
        },
                Mode = "payment",
                PaymentIntentData = new SessionPaymentIntentDataOptions
                {
                    Metadata = new Dictionary<string, string>
            {
                { "wallet_funds", amount.ToString() },
                { "user_id", userId },
            },
                },
                SuccessUrl =
$"https://gamesino.azurewebsites.net/Payments/AddFundsSuccess?session_id={
{CHECKOUT_SESSION_ID}}&user_id={WebUtility.UrlEncode(userId)}&added_amount
={WebUtility.UrlEncode(amount.ToString(CultureInfo.InvariantCulture))}",
                CancelUrl =
"https://gamesino.azurewebsites.net/Payments/Cancel",
            };

            var service = new SessionService();
            Session session = service.Create(options);

            Response.Headers.Add("Location", session.Url);
            return new StatusCodeResult(303);
        }


        [Authorize]
        public IActionResult Success()
        {
            var items = _shoppingCart.GetShoppingCartItems();
            string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
            string firstName = "";
            string lastName = "";
```

```
            string addressLine1 = "";
            string addressLine2 = "";
            string zipCode = "";
            string city = "";
            string country = "";
            string phoneNumber = "";
            string email = "";
            int orderId = 0;
            int amount = 0;

                // If there's no addedAmount query parameter, create the
order.
                string sessionId =
HttpContext.Request.Query["session_id"];
                _orderRepository.CreateOrder(items, userId, firstName,
lastName, addressLine1, addressLine2, zipCode, city, country, phoneNumber,
email);
                _shoppingCart.ClearCart();

            return View();
        }


        [Authorize]
        public IActionResult AddFundsSuccess()
        {
            // Get the user ID and added amount from the query parameters
            string userId = HttpContext.Request.Query["user_id"];
            string addedAmountString =
HttpContext.Request.Query["added_amount"];

            if (!string.IsNullOrEmpty(addedAmountString))
            {
                decimal parsedAddedAmount =
Convert.ToDecimal(addedAmountString);
                _walletRepository.AddFundsToWallet(userId,
parsedAddedAmount);
            }

            return View();
```

```
        }

        [Authorize]
        public IActionResult Cancel()
        {
            return View();
        }

        [Authorize]
        public IActionResult Index()
        {
            _shoppingCart.ShoppingCartItems =
_shoppingCart.GetShoppingCartItems();
            var orderSummaryViewModel = new OrderSummaryViewModel
            {
                ShoppingCart = _shoppingCart,
                ShoppingCartTotal = _shoppingCart.GetShoppingCartTotal(),
            };

            return View(orderSummaryViewModel);
        }
    }
}
```

PlayGameController

```
using GamesPlatform.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class PlayGameController : Controller
    {
        private readonly IWalletRepository _walletRepository;
```

```
        public PlayGameController(IWalletRepository walletRepository)
        {
            _walletRepository = walletRepository;
        }
        public IActionResult PlayGame(string gameID)
        {
            ViewBag.GameID = gameID;
            return View();
        }

        [Authorize]
        [HttpPost]
        public IActionResult SubtractFromWalletAndPlayGame(string gameID)
        {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            decimal amountToSubtract = 5.00M;
            decimal userBalance = _walletRepository.GetBalance(userId);
            if (userBalance < amountToSubtract)
            {
                TempData["ErrorMessage"] = "Add some funds to your account
in order to play this game.";
                return RedirectToAction("Index", "Casino");
            }


            _walletRepository.SubtractFromWallet(userId,
amountToSubtract);
            return RedirectToAction("PlayGame", new { GameID = gameID });
        }
    }
}
```

RPSController

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
```

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class RPSController : Controller
    {
        private readonly IGamesRepository _gamesRepository;
        private readonly ICasinoResultRepository _casinoResultRepository;
        private readonly IWalletRepository _walletRepository;

        public RPSController(IGamesRepository gamesRepository,
ICasinoResultRepository casinoResultRepository, IWalletRepository
walletRepository)
        {
            _gamesRepository = gamesRepository;
            _casinoResultRepository = casinoResultRepository;
            _walletRepository = walletRepository;
        }
        [Authorize]
        public IActionResult RPSIndex()
        {
            return RedirectToAction("Play", "RPS");
        }

        [Authorize]
        public IActionResult Play(string userChoice)
        {
            if (userChoice == null)
            {
                return View("RPSIndex");
            }

            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var computerChoice = GetComputerChoice();
            var (result, amountWon) = DetermineResult(userChoice,
computerChoice);
```

```
            var win = result == "Congratulations! You won!";

            var gameResult = new CasinoResult
            {
                UserChoice = userChoice,
                ComputerChoice = computerChoice,
                Result = result,
                Win = win,
                AmountWon = amountWon,
                DateResultPlaced = DateTime.UtcNow
            };
            _casinoResultRepository.CreateCasinoResult(gameResult);
            if (gameResult.Win)
            {
                _walletRepository.AddToWallet(userId,
gameResult.AmountWon);
            }
            return View("RPSIndex", gameResult);
        }

        private string GetComputerChoice()
        {
            var random = new Random();
            int choice = random.Next(1, 4);

            return choice switch
            {
                1 => "Rock",
                2 => "Paper",
                _ => "Scissors"
            };
        }

        private (string, decimal) DetermineResult(string userChoice,
string computerChoice)
        {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            if (userChoice == computerChoice) return ("It's a tie!", 0);
```

```
            if (userChoice == "Rock" && computerChoice == "Scissors")
return ("Congratulations! You won!", 10);
            if (userChoice == "Paper" && computerChoice == "Rock") return
("Congratulations! You won!", 10);
            if (userChoice == "Scissors" && computerChoice == "Paper")
return ("Congratulations! You won!", 10);
            return ("Sorry, you lost!", 0);
        }
    }
}
```

ShoppingCartController

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class ShoppingCartController : Controller
    {
        private readonly IGamesRepository _gamesRepository;
        private readonly ShoppingCart _shoppingCart;
        public ShoppingCartController(IGamesRepository gamesRepository,
ShoppingCart shoppingCart)
        {
            _gamesRepository= gamesRepository;
            _shoppingCart= shoppingCart;
        }
        [Authorize]
        public ViewResult Index()
        {
            var items= _shoppingCart.GetShoppingCartItems();
```

```
            // Check if the shopping cart is empty and store it in ViewBag
            ViewBag.IsEmpty = (items.Count == 0);

            _shoppingCart.ShoppingCartItems = items;

            var sCVM = new ShoppingCartViewModel
            {
                ShoppingCart = _shoppingCart,
                ShoppingCartTotal = _shoppingCart.GetShoppingCartTotal()
            };
            return View(sCVM);
        }
        [Authorize]
        public RedirectToActionResult AddToShoppingCart(int gameID)
        {
            var selectedGame = _gamesRepository.Games.FirstOrDefault(p =>
p.GameID == gameID);
            if(selectedGame != null)
            {
                _shoppingCart.AddToCart(selectedGame, 1);
            }
            return RedirectToAction("Index");
        }
        [Authorize]
        public RedirectToActionResult RemoveFromShoppingCart(int gameID)
        {
            var selectedGame = _gamesRepository.Games.FirstOrDefault(p =>
p.GameID == gameID);
            if (selectedGame != null)
            {
                _shoppingCart.RemoveFromCart(selectedGame);
            }
            return RedirectToAction("Index");
        }

        public RedirectToActionResult ClearCart()
        {
            _shoppingCart.ClearCart();
            return RedirectToAction("Index");
        }
```

```
    }
}
```

SlotMachineController

```
using GamesPlatform.Repositories;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class SlotMachineController : Controller
    {
        private readonly ApplicationDbContext _applicationDbContext;
        private readonly IWalletRepository _walletRepository;

        public SlotMachineController(ApplicationDbContext
applicationDbContext, IWalletRepository walletRepository)
        {
            _applicationDbContext = applicationDbContext;
            _walletRepository = walletRepository;
        }
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Spin(SlotMachineViewModel model)
        {
            var rand = new Random();
```

```
            string[] symbols = { "A", "B", "C", "7" }; // Add or modify
the symbols as desired

            model.Reels = new string[3];
            for (int i = 0; i < 3; i++)
            {
                model.Reels[i] = symbols[rand.Next(symbols.Length)];
            }

            model.IsWinner = model.Reels[0] == model.Reels[1] &&
model.Reels[1] == model.Reels[2];
            model.AmountWon = model.IsWinner.Value ? 10 : 0; // Change
this to the desired win amount

            var casinoResult = new CasinoResult
            {
                UserChoice = string.Join(",", model.Reels),
                ComputerChoice = string.Join(",", model.Reels),
                Result = model.IsWinner.Value ? "Win" : "Lose",
                Win = model.IsWinner.Value,
                AmountWon = model.AmountWon,
                DateResultPlaced = DateTime.Now
            };

            _applicationDbContext.CasinoResults.Add(casinoResult);
            await _applicationDbContext.SaveChangesAsync();

            if (model.IsWinner.Value)
            {
                string userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
                _walletRepository.AddToWallet(userId, model.AmountWon);
            }

            model.CasinoResult = casinoResult;

            return View("Index", model);
        }
    }
}
```

VerificationController

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class VerificationController : Controller
    {
        private readonly IVerificationRepository _verificationRepository;
        public VerificationController(IVerificationRepository
verificationRepository)
        {
            _verificationRepository = verificationRepository;
        }

        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public IActionResult Index(FileUpload fileObject)
        {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            var verificationExists =
_verificationRepository.RetrieveVerificationByUserID(userId);
            if (verificationExists == null)
            {
                if (fileObject.file.Length > 0)
```

```
                {
                    using (var memoryStream = new MemoryStream())
                    {
                        fileObject.file.CopyTo(memoryStream);
                        if (memoryStream.Length < 2097152)
                        {
                            var verificationRequest = new Verification()
                            {
                                Content = memoryStream.ToArray(),
                                UserID = userId,
                                Status = "Pending"

                            };

_verificationRepository.CreateVerification(verificationRequest);
                            ViewBag.SuccessMessage = "Thank you for
submitting your ID to get access to casino games. Estimated wait time is
1-2 days so check back later  ";
                        }
                        else
                        {
                            ModelState.AddModelError("File", "The file is
too large");
                        }
                    }
                }
            }
            else if (verificationExists.Status == "Approved")
            {
                ViewBag.ErrorMessage = "YThis account is approved already
and allowed to play Casino Games.";
            }
            else if (verificationExists.Status == "Denied")
            {
                ViewBag.ErrorMessage = "Your Verification Request has been
rejected by Gamesino Admins. Email support if you think this is an
mistake.";
            }
            else if (verificationExists.Status == "Pending")
            {
```

```
                ViewBag.ErrorMessage = "Your Verification request is
currently being reviewed by Gamesino Admins. Check back Later";
            }
            else if (verificationExists.Status == "")
            {
                ViewBag.ErrorMessage = "There seems to be an issue with
your request. Contact support as soon as yyou can to resolve this.";
            }

            return View();
        }
    }
}
```

WalletController

```
using GamesPlatform.Interfaces;
using GamesPlatform.ViewModels;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Runtime.CompilerServices;
using System.Security.Claims;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class WalletController : Controller
    {
        private readonly IWalletRepository _walletRepository;
        public WalletController(IWalletRepository walletRepository)
        {
            _walletRepository = walletRepository;
        }
        public ViewResult Index()
        {
            string userId =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
```

```
            decimal balance = _walletRepository.GetBalance(userId);

            var walletViewModel = new WalletViewModel
            {
                Balance = balance
            };

            return View(walletViewModel);
        }
    }
}
```

WithdrawController

```
using Microsoft.AspNetCore.Mvc;
using PayoutsSdk.Payouts;
using PayoutsSdk.Core;
using PayPalHttp;
using Microsoft.AspNetCore.Authorization;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using System.Security.Claims;
using System.Globalization;
using Azure.Core;
using PayPal.Api;
using HttpResponse = PayPalHttp.HttpResponse;

namespace GamesPlatform.Controllers
{
    [Authorize]
    public class WithdrawController : Controller
    {
        private readonly IWithdrawRepository _withdrawRepository;
        private readonly IWalletRepository _walletRepository;
        public WithdrawController(IWithdrawRepository withdrawRepository,
IWalletRepository walletRepository)
```

```
        {
            _withdrawRepository = withdrawRepository;
            _walletRepository = walletRepository;
        }

        [Authorize]
        public IActionResult Index()
        {
            return View();
        }
        [Authorize]
        [HttpPost]
        public IActionResult Index(Withdraw withdrawMade)
        {
            string userID =
User.FindFirst(ClaimTypes.NameIdentifier).Value;
            decimal walletBalance = _walletRepository.GetBalance(userID);
            int walletId = _walletRepository.RetrieveWalletID(userID);
            if (walletId == 0)
            {
                ViewBag.ErrorMessage = "Add Some Funds to your account";
                return View();
            }
            withdrawMade.WalletID = walletId;
            if (withdrawMade.AmountTransfered < walletBalance &&
withdrawMade.AmountTransfered >= 5.00M)

            {
                string amountTransferred =
withdrawMade.AmountTransfered.ToString();
                string email = withdrawMade.PayPalEmail;
                var response = CreatePayout(amountTransferred, email);
                HttpResponse createPayoutResponse = response.Result;
                var withdraw =
createPayoutResponse.Result<CreatePayoutResponse>();
                withdrawMade.PayPalBatchID =
withdraw.BatchHeader.PayoutBatchId;
                _walletRepository.SubtractFromWallet(userID,
withdrawMade.AmountTransfered);
                _withdrawRepository.WithdrawMade(withdrawMade);
```

```
                ViewBag.SuccessMessage = "You have successfully transfered
€" + amountTransferred + " to your PayPal account";

            }
            else
            {
                ViewBag.ErrorMessage = "We were unable to transfer funds
into the PayPal account with the email provided. Do you have enough in
your wallet?";
            }
            return View();
        }

        private static CreatePayoutRequest buildRequstbody(string
amountTransferred, string email)
        {
            var request = new CreatePayoutRequest()
            {
                SenderBatchHeader = new SenderBatchHeader
                {
                    EmailMessage = "You have received a payout from
Gamesino!: €" + amountTransferred,
                    EmailSubject = $"Gamesino Money Transferred
Successfully!"
                },
                Items = new List<PayoutsSdk.Payouts.PayoutItem>(){
                    new PayoutsSdk.Payouts.PayoutItem(){
                        RecipientType="EMAIL",

                        Amount=new PayoutsSdk.Payouts.Currency(){
                            CurrencyCode="EUR",
                            Value=amountTransferred,
                         },
                        Receiver=email,

                    }
                }
            };
            return request;
        }
```

```
        public async static Task<HttpResponse> CreatePayout(string
amountTransfered, string email)
        {
            Console.WriteLine("Creating payout with complete payload");

            try
            {
                PayoutsPostRequest request = new PayoutsPostRequest();
                request.RequestBody(buildRequstbody(amountTransfered,
email));
                var response = await
PayPalClient.client().Execute(request);
                var result = response.Result<CreatePayoutResponse>();
                return response;
            }
            catch (HttpException ex)
            {
                return null;
            }
        }
    }
}
```

Data

ApplicationDbContext.cs

```
namespace GamesPlatform.Data
{
    public class ApplicationDbContext : IdentityDbContext<IdentityUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
```

```
            : base(options)
        {
        }
        public DbSet<Game> Games { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<ShoppingCartItem> ShoppingCartItems { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderDetail> OrderDetails { get; set; }
        public DbSet<CasinoResult> CasinoResults { get; set; }
        public DbSet<Wallet> Wallets { get; set; }
        public DbSet<Withdraw> Withdraws { get; set; }
        public DbSet<Verification> Verifications { get; set; }
    }
}
```

StripeSettings.cs

```
namespace GamesPlatform.Data
{
    public class StripeSettings
    {
        public string PublishableKey { get; set; }
        public string SecretKey { get; set; }
    }
}
```

Interfaces

ICasinoResultRepository.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.Interfaces
{
    public interface ICasinoResultRepository
```

```
    {
        void CreateCasinoResult(CasinoResult result);
    }
}
```

ICategoryRepository.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.Interfaces
{
    public interface ICategoryRepository
    {
        IEnumerable<Category> Categories { get; }
    }
}
```

IGamesRepository.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.Interfaces
{
    public interface IGamesRepository
    {
        IEnumerable<Game> Games { get;}
        IEnumerable<Game> FeaturedGames { get;}
        IEnumerable<Game> FreeGames { get; }
        IEnumerable<Game> CasinoGames { get; }
        Game GetGamesById (int GamesId);
    }
}
```

IOrderDetailRepository.cs

```
using GamesPlatform.Models;
using Microsoft.AspNetCore.Http.Features;

namespace GamesPlatform.Interfaces
{
    public interface IOrderDetailRepository
    {
        List<OrderDetail> GetAllOrderDetails(int orderID);
        OrderDetail GetOrderDetailById(int orderDetailID);
    }
}
```

IOrderRepository.cs

```
using GamesPlatform.Models;
using Microsoft.AspNetCore.Http.Features;

namespace GamesPlatform.Interfaces
{
    public interface IOrderRepository
    {
        void CreateOrder(List<ShoppingCartItem> items, string userId,
string firstName, string lastName, string addressLine1, string
addressLine2, string zipCode, string city, string country, string
phoneNumber, string email);
        List<Order> GetOrdersByUserID(string userId);

    }
}
```

IVerificationRepository.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.Interfaces
{
    public interface IVerificationRepository
    {
        void CreateVerification(Verification verification);
```

```
        void UpdateVerification(int verificationId, string status);
        IEnumerable<Verification> RetrieveAllVerifications();
        Verification RetrieveVerificationByID(int verificationId);
        Verification RetrieveVerificationByUserID(string userId);
    }
}
```

IWalletRepository.cs

```
namespace GamesPlatform.Interfaces
{
    public interface IWalletRepository
    {
        void AddToWallet(string userId, decimal addedAmount);
        void SubtractFromWallet(string userId, decimal removedAmount);
        decimal GetBalance(string userId);
        int RetrieveWalletID(string userId);
        void AddFundsToWallet(string userId, decimal amount);
    }
}
```

IWithdrawRepository.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.Interfaces
{
    public interface IWithdrawRepository
    {
        void WithdrawMade(Withdraw withdraw);
    }
}
```

Mocks

MockCategoryRepository.cs

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Mocks
{
    public class MockCategoryRepository : ICategoryRepository
    {
        public IEnumerable<Category> Categories => new List<Category>
        {
            new Category{CategoryID=1, CategoryName="Web Games",
CategoryDescription= "Available games you can play on your browser"},
            new Category{CategoryID=2, CategoryName="Mobile Games",
CategoryDescription= "Avaliable games to play on your mobile device"},
            new Category{CategoryID=3, CategoryName="Casino Games",
CategoryDescription= "Test your luck in these casino ga mes"},
            new Category{CategoryID=4, CategoryName="Free Games",
CategoryDescription= "Enjoy these free games hand-picked by Gamesino!"}
        };
    }
}
```

MockGamesRepository.cs

```
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using static System.Net.WebRequestMethods;

namespace GamesPlatform.Mocks
{
    public class MockGamesRepository : IGamesRepository
    {
        private readonly ICategoryRepository _categoryRepository = new
MockCategoryRepository();
        public IEnumerable<Game> Games => new List<Game>
            {
                new Game{
```

```
                    GameID=1,
                    GameName="Kick Ups",
                    Category = _categoryRepository.Categories.First(),
                    GameDescription= "The goal is simple, keep the ball in
the air for the longest amount of time...",
                    Price=20.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = true
                },


                new Game{
                    GameID=2,
                    GameName="Amoeba Online",
                    Category = _categoryRepository.Categories.First(),
                    GameDescription= "Jump into the ultimate Single cell
battle! Compete aganist 100 players or bots in this multiplayer game!",
                    Price=10.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = true
                },

                new Game{
                    GameID=3,
                    GameName="Reaction Test Game",
                    Category = _categoryRepository.Categories.First(),
                    GameDescription= "Test your reaction time and compare
with others",
                    Price=50.00M,
```

```
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = true
                },

                new Game{
                    GameID=4,
                    GameName="Dungeon Deathmatch",
                    Category = _categoryRepository.Categories.First(),
                    GameDescription= "Dunegon Deathmatch is a fast paced
top down multiplayer deathmatch game. Fight up to 8 players in this action
game!",
                    Price=20.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = false
                },

                new Game{
                    GameID=5,
                    GameName="Dave Wakes Up",
                    GameDescription= "Dave wakes up is a short narrative
top down 2d game. You play as Dave, trapped in a corpo world!",
                    Price=20.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = false
                },
```

```
                new Game{
                    GameID=6,
                    GameName="Tower Of Hanoi",
                    GameDescription= "Objective of the pizzle is to move
the entire stack to another rod one at a time.",
                    Price=0.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = false,
                    IsFreeGame=true
                },

                new Game{
                    GameID=7,
                    GameName="High or Low",
                    GameDescription= "",
                    Price=0.00M,
                    GameImageUrl
="https://i2-prod.manchestereveningnews.co.uk/sport/football/article249854
59.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    GameImageThumbnailUrl =
"https://i2-prod.manchestereveningnews.co.uk/sport/football/article2498545
9.ece/ALTERNATES/s615/0_GettyImages-1413133364.jpg",
                    IsFeaturedGame = false,
                    IsFreeGame=true
                },
            };

        public IEnumerable<Game> FeaturedGames { get; }
        public IEnumerable<Game> FreeGames { get; }
        public IEnumerable<Game> CasinoGames { get; }


        public Game GetGamesById(int GameId)
        {
            throw new NotImplementedException();
```

```
        }
    }
}
```

Models

Applicationuser.cs

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.Models
{
    public class ApplicationUser: IdentityUser
    {
        [Display(Name = "Full Name")]
        public string FullName { get; set; }
        public DateTime DateOfBirth { get; set; }
    }
}
```

CasinoResult.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.Models
{
    public class CasinoResult
    {
            [Key]
            [BindNever]
            public int ResultID { get; set; }
            public string UserChoice { get; set; }
            public string ComputerChoice { get; set; }
            public string Result { get; set; }
            public bool Win { get; set; }
```

```
            public decimal AmountWon { get; set; }
            public DateTime DateResultPlaced { get; set; }

    }
}
```

Category.cs

```
namespace GamesPlatform.Models
{
    public class Category
    {
        public int CategoryID { get; set; }
        public string CategoryName { get; set; }
        public string CategoryDescription { get; set; }
        public  List<Game> Games { get; set; }
    }
}
```

ErrorViewModel.cs

```
namespace GamesPlatform.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

FileUpload.cs

```
using PayoutsSdk.Payouts;
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.Models
{
    public class FileUpload
```

```
    {
        [Required]
        [Display(Name = "File")]
        public IFormFile file { get; set; }
    }
}
```

Game.cs

```
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.Models
{
    public class Game
    {
        public int GameID { get; set; }
        public string GameName { get; set; }
        public string GameDescription { get; set; }
        [Range(0.00, 900.00)]
        public decimal Price { get; set; }
        public string GameImageUrl { get; set; }
        public string GameVideoUrl { get; set; }
        public string GameImageThumbnailUrl { get; set; }
        public bool IsFeaturedGame { get; set; }
        public bool IsFreeGame { get; set; }
        public bool IsCasinoGame { get; set; }
        public int CategoryID { get; set; }
        public Category Category { get; set; }

    }
}
```

Order.cs

```
using System.ComponentModel.DataAnnotations;
using System.Drawing;
using System.Numerics;
```

```
namespace GamesPlatform.Models
{
    public class Order
    {
        public int OrderID { get; set; }
        public string UserID { get; set; }

        [Display(Name = "FirstName")]
        [StringLength(50)]
        [Required(ErrorMessage ="Please enter your First Name please!")]
        public string FirstName { get; set; }

        [Required(ErrorMessage = "Please enter your Last Name please!")]
        [Display(Name = "Last Name")]
        [StringLength(50)]
        public string LastName { get; set; }

        [Required(ErrorMessage = "Please enter your Address")]
        [Display(Name = "Address")]
        [StringLength(100)]
        public string AddressLine1 { get; set; }

        [Required(ErrorMessage = "Please enter your Address 2 please!")]
        [Display(Name = "Address2")]
        [StringLength(100)]
        public string AddressLine2 { get; set;}

        [Required(ErrorMessage = "Please enter your ZipCode please!")]
        [Display(Name = "ZipCode")]
        [StringLength(20)]
        public string ZipCode { get; set; }

        [Required(ErrorMessage = "Please enter your City please!")]
        [Display(Name = "City")]
        [StringLength(50)]
        public string City { get; set; }

        [Required(ErrorMessage = "Please enter your Country please!")]
        [Display(Name = "Country")]
```

```
        [StringLength(50)]
        public string Country { get; set; }

        [Required(ErrorMessage = "Please enter your PhoneNumber please!")]
        [DataType(DataType.PhoneNumber)]
        [Display(Name = "PhoneNumber")]
        [StringLength(50)]
        public string PhoneNumber { get; set; }

        [Required(ErrorMessage = "Please enter your Email please!")]
        [Display(Name = "Email")]
        [DataType(DataType.EmailAddress)]

[RegularExpression(@"(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=
?^_`{|}~-]+)*|""(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\
x01-\x09\x0b\x0c\x0e-\x7f])*"")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[
a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]
?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x0
1-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])
+)\])",
        ErrorMessage = "The email address is not entered in a correct
format")]
        [StringLength(50)]
        public string Email { get; set; }

        public decimal OrderTotal { get; set; }
        public DateTime OrderPlaced { get; set; }
        public List<OrderDetail> OrderDetails { get; set; }
    }
}
```

OrderDetail.cs

```
namespace GamesPlatform.Models
{
    public class OrderDetail
    {
        public int OrderDetailID { get; set; }
        public int OrderID { get; set; }
        public int GameID { get; set; }
```

```
        public int Amount { get; set; }
        public decimal Price { get; set; }
        public Game Game { get; set; }
        public Order Order { get; set; }
    }
}
```

PaginatedList.cs

```
using Microsoft.EntityFrameworkCore;

namespace GamesPlatform.Models
{
    public class PaginatedList<T> : List<T>
    {
        public int PageIndex { get; private set; }
        public int TotalPages { get; private set; }

        public PaginatedList(List<T> items, int count, int pageIndex, int
pageSize)
        {
            PageIndex = pageIndex;
            TotalPages = (int)Math.Ceiling(count / (double)pageSize);

            this.AddRange(items);
        }

        public bool HasPreviousPage
        {
            get
            {
                return (PageIndex > 1);
            }
        }

        public bool HasNextPage
        {
            get
```

```
            {
                return (PageIndex < TotalPages);
            }
        }

        public static async Task<PaginatedList<T>>
CreateAsync(IQueryable<T> source, int pageIndex, int pageSize)
        {
            var count = await source.CountAsync();
            var items = await source.Skip((pageIndex - 1) *
pageSize).Take(pageSize).ToListAsync();
            return new PaginatedList<T>(items, count, pageIndex,
pageSize);
        }
    }
}
```

ShoppingCart.cs

```
using GamesPlatform.Data;
using Microsoft.EntityFrameworkCore;

namespace GamesPlatform.Models
{
    public class ShoppingCart
    {
        private readonly ApplicationDbContext _applicationDbContext;
        private ShoppingCart(ApplicationDbContext applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }

        public string ShoppingCartID { get; set; }
        public List<ShoppingCartItem> ShoppingCartItems { get; set; }

        public static ShoppingCart GetCart(IServiceProvider services)
        {
            ISession session =
services.GetRequiredService<IHttpContextAccessor>()?
            .HttpContext.Session;
```

```
            var context = services.GetService<ApplicationDbContext>();
            string cartID = session.GetString("CartID") ??
Guid.NewGuid().ToString();

            session.SetString("CartID", cartID);

            return new ShoppingCart(context) { ShoppingCartID = cartID };
        }
        public void AddToCart(Game game, int amount)
        {
            var shoppingCartItem =

_applicationDbContext.ShoppingCartItems.SingleOrDefault(s => s.Game.GameID
== game.GameID && s.ShoppingCartID == ShoppingCartID);

            if (shoppingCartItem == null)
            {
                shoppingCartItem = new ShoppingCartItem
                {
                    ShoppingCartID = ShoppingCartID,
                    Game = game,
                    Amount = 1
                };


_applicationDbContext.ShoppingCartItems.Add(shoppingCartItem);
            }
            else
            {
                shoppingCartItem.Amount++;
            }
            _applicationDbContext.SaveChanges();
        }

        public int RemoveFromCart(Game game)
        {
            var shoppingCartItem =

_applicationDbContext.ShoppingCartItems.SingleOrDefault(
```

```
                        s => s.Game.GameID == game.GameID &&
s.ShoppingCartID == ShoppingCartID);

            var localAmount = 0;

            if (shoppingCartItem != null)
            {
                if (shoppingCartItem.Amount > 1)
                {
                    shoppingCartItem.Amount--;
                    localAmount = shoppingCartItem.Amount;
                }
                else
                {

_applicationDbContext.ShoppingCartItems.Remove(shoppingCartItem);
                }
            }

            _applicationDbContext.SaveChanges();

            return localAmount;
        }

        public List<ShoppingCartItem> GetShoppingCartItems()
        {
            return ShoppingCartItems ??
                   (ShoppingCartItems =
                       _applicationDbContext.ShoppingCartItems.Where(c =>
c.ShoppingCartID == ShoppingCartID)
                           .Include(s => s.Game)
                           .ToList());
        }

        public void ClearCart()
        {
            var cartItems = _applicationDbContext
                .ShoppingCartItems
                .Where(cart => cart.ShoppingCartID == ShoppingCartID);
```

```
_applicationDbContext.ShoppingCartItems.RemoveRange(cartItems);

            _applicationDbContext.SaveChanges();
        }

        public decimal GetShoppingCartTotal()
        {
            var total = _applicationDbContext.ShoppingCartItems.Where(c =>
c.ShoppingCartID == ShoppingCartID)
                .Select(c => c.Game.Price * c.Amount).Sum();
            return total;
        }

    }
}
```

ShoppingCartitem.cs

```
namespace GamesPlatform.Models
{
    public class ShoppingCartItem
    {

        public int  ShoppingCartItemID { get; set; }
        public Game Game { get; set; }
        public int Amount { get; set; }
        public string ShoppingCartID { get; set; }

    }
}
```

Verification.cs

```
namespace GamesPlatform.Models
{
    public class Verification
    {
        public int VerificationID { get; set; }
```

```
        public string UserID { get; set; }
        public byte[] Content { get; set; }
        public string Status { get; set; }
        public DateTime DateOfRequest { get; set; }

    }
}
```

Wallet.cs

```
namespace GamesPlatform.Models
{
    public class Wallet
    {
        public int WalletID { get; set; }
        public string UserID { get; set; }
        public decimal WalletBalance { get; set; }
    }
}
```

Withdraw.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding;
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.Models
{
    public class Withdraw
    {
        [BindNever]
        public int WithdrawID { get; set; }
        public int WalletID { get; set; }

        [Required(ErrorMessage = "Please enter in the amount you want to
transfer between 5-500")]
        [Display(Name = "Transfer Amount")]
        [DataType(DataType.Currency)]
        [Range(5, 500)]
        public decimal AmountTransfered { get; set; }
```

```
        [Required(ErrorMessage = "Required field")]
        [Display(Name = "PayPal Email")]
        [EmailAddress]
        public string PayPalEmail { get; set; }
        public string PayPalBatchID { get; set; }
        public DateTime WithdrawDate { get; set; }
    }
}
```

Repositories

CasionoResultRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Repositories
{
    public class CasinoResultRepository : ICasinoResultRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        public CasinoResultRepository(ApplicationDbContext
applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }
        public void CreateCasinoResult(CasinoResult gameResult)
        {

            gameResult.DateResultPlaced = DateTime.Now;
            _applicationDbContext.Add(gameResult);
            _applicationDbContext.SaveChanges();
        }
    }
}
```

CategoryRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Repositories
{
    public class CategoryRepository : ICategoryRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        public CategoryRepository(ApplicationDbContext
applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }

        public IEnumerable<Category> Categories =>
_applicationDbContext.Categories;
    }
}
```

GameRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using Microsoft.EntityFrameworkCore;

namespace GamesPlatform.Repositories
{
    public class GameRepository : IGamesRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        public GameRepository(ApplicationDbContext applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }
```

```
        public IEnumerable<Game> Games =>
_applicationDbContext.Games.Include(c => c.Category);

        public IEnumerable<Game> FeaturedGames =>
_applicationDbContext.Games.Where(p => p.IsFeaturedGame).Include(c =>
c.Category);

        public IEnumerable<Game> FreeGames =>
_applicationDbContext.Games.Where(p => p.IsFreeGame).Include(c =>
c.Category);
        public IEnumerable<Game> CasinoGames =>
_applicationDbContext.Games.Where(p => p.IsCasinoGame).Include(c =>
c.Category);

        public Game GetGamesById(int GamesId) =>
_applicationDbContext.Games.FirstOrDefault(p => p.GameID == GamesId);

    }
}
```

OrderRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json.Bson;
using System.Diagnostics.Metrics;
using System.Reflection.Emit;

namespace GamesPlatform.Repositories
{
    public class OrderRepository : IOrderRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        private readonly ShoppingCart _shoppingCart;

        public OrderRepository(ApplicationDbContext applicationDbContext,
ShoppingCart shoppingCart)
        {
```

```
            _applicationDbContext = applicationDbContext;
            _shoppingCart = shoppingCart;
        }

        public void CreateOrder(List<ShoppingCartItem> items, string
userId, string firstName, string lastName, string addressLine1, string
addressLine2, string zipCode, string city, string country, string
phoneNumber, string email)
        {
            //order.OrderPlaced = DateTime.Now;
            //_applicationDbContext.Orders.Add(order);

            var order = new Order()
            {
                UserID = userId,
                FirstName = firstName,
                LastName = lastName,
                AddressLine1 = addressLine1,
                AddressLine2 = addressLine2,
                ZipCode = zipCode,
                City = city,
                Country = country,
                PhoneNumber = phoneNumber,
                Email = email


            };
            _applicationDbContext.Orders.Add(order);
            _applicationDbContext.SaveChanges();

            var shoppingCartItems = _shoppingCart.ShoppingCartItems;

            foreach (var item in shoppingCartItems)
            {
                var orderDetail = new OrderDetail()
                {

                    Amount = item.Amount,
                    GameID = item.Game.GameID,
                    OrderID = order.OrderID,
```

```
                    Price = item.Game.Price,

                };
                _applicationDbContext.OrderDetails.Add(orderDetail);
            }
            _applicationDbContext.SaveChanges();
        }


        public List<Order> GetOrdersByUserID(string userId)
        {
            var orders = _applicationDbContext.Orders.Include(n =>
n.OrderDetails).ThenInclude(n => n.Game).Where(n => n.UserID ==
userId).ToList();
            return orders;
        }
    }
}
```

VerificationRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Repositories
{
    public class VerificationRepository : IVerificationRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        public VerificationRepository(ApplicationDbContext
applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }
        public void CreateVerification(Verification verification)
        {
            verification.DateOfRequest = DateTime.Now;
            _applicationDbContext.Verifications.Add(verification);
```

```
            _applicationDbContext.SaveChanges();
        }

        public IEnumerable<Verification> RetrieveAllVerifications()
        {
            var result = (from v in _applicationDbContext.Verifications
                          select v).ToList();
            return result;
        }

        public Verification RetrieveVerificationByID(int verificationId)
        {
            _applicationDbContext.Verifications.FirstOrDefault(w =>
w.VerificationID == verificationId);
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.VerificationID
== verificationId);
            return result;
        }

        public Verification RetrieveVerificationByUserID(string userId)
        {
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.UserID ==
userId);
            return result;
        }

        public void UpdateVerification(int verificationId, string status)
        {
            _applicationDbContext.Verifications.FirstOrDefault(w =>
w.VerificationID == verificationId);
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.VerificationID
== verificationId);

            result.Status = status;

            _applicationDbContext.SaveChanges();
        }
```

```
    }
}
```

WalletRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Repositories
{
    public class VerificationRepository : IVerificationRepository
    {
        private readonly ApplicationDbContext _applicationDbContext;
        public VerificationRepository(ApplicationDbContext
applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }
        public void CreateVerification(Verification verification)
        {
            verification.DateOfRequest = DateTime.Now;
            _applicationDbContext.Verifications.Add(verification);
            _applicationDbContext.SaveChanges();
        }

        public IEnumerable<Verification> RetrieveAllVerifications()
        {
            var result = (from v in _applicationDbContext.Verifications
                          select v).ToList();
            return result;
        }

        public Verification RetrieveVerificationByID(int verificationId)
        {
            _applicationDbContext.Verifications.FirstOrDefault(w =>
w.VerificationID == verificationId);
```

```
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.VerificationID
== verificationId);
            return result;
        }

        public Verification RetrieveVerificationByUserID(string userId)
        {
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.UserID ==
userId);
            return result;
        }

        public void UpdateVerification(int verificationId, string status)
        {
            _applicationDbContext.Verifications.FirstOrDefault(w =>
w.VerificationID == verificationId);
            var result =
_applicationDbContext.Verifications.FirstOrDefault(w => w.VerificationID
== verificationId);

            result.Status = status;

            _applicationDbContext.SaveChanges();
        }
    }
}
```

WithdrawRepository.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Models;

namespace GamesPlatform.Repositories
{
    public class WithdrawRepository : IWithdrawRepository
    {
```

```
        private readonly ApplicationDbContext _applicationDbContext;
        public WithdrawRepository(ApplicationDbContext
applicationDbContext)
        {
            _applicationDbContext = applicationDbContext;
        }
        public void WithdrawMade(Withdraw withdraw)
        {
            withdraw.WithdrawDate = DateTime.Now;

            _applicationDbContext.Withdraws.Add(withdraw);

            _applicationDbContext.SaveChanges();
        }
    }
}
```

TagHelpers

EmailTagHelper.cs

```
using Microsoft.AspNetCore.Razor.TagHelpers;
using System.Drawing;

namespace GamesPlatform.TagHelpers
{
    public class EmailTagHelper : TagHelper
    {
        public string Address { get; set; }
        public string Content { get; set; }

        public override void Process(TagHelperContext context,
TagHelperOutput output)

        {
            output.TagName = "a";
            output.Attributes.SetAttribute("href", "mailto:" + Address);
            output.Content.SetContent(Content);
```

```
        }
    }
}
```

ViewModels

BlackJackViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class BlackJackViewModel
    {
        public List<int> PlayerHand { get; set; }
        public List<int> DealerHand { get; set; }
        public bool? IsWinner { get; set; }
        public decimal AmountWon { get; set; }
        public CasinoResult CasinoResult { get; set; }
    }
}
```

CasinoViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class CasinoViewModel
    {
        public IEnumerable<Game> CasinoGame { get; set; }
    }
}
```

CoinFlipViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class CoinFlipViewModel
    {
        public string PlayerChoice { get; set; }
        public string Result { get; set; }
        public bool? IsWinner { get; set; }
        public CasinoResult CasinoResult { get; set; }
    }
}
```

CreateRoleViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace GamesPlatform.ViewModels
{
    public class CreateRoleViewModel
    {
        [Required]
        public string RoleName { get; set;}
    }
}
```

GameListViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class GameListViewModel
    {
        public IEnumerable<Game> Games { get; set; }
        public string CurrentCategory { get; set; }
        public IEnumerable<int> PurchasedGameIDs { get; set; }
    }
}
```

HomeViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class HomeViewModel
    {public IEnumerable<Game> FeaturedGames { get; set; }
        public IEnumerable<Game> FreeGames { get; set; }
        public IEnumerable<int> PurchasedGameIDs { get; set; }
    }
}
```

LibraryViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class LibraryViewModel
    {
        public List<Game> PurchasedGames { get; set; }
    }
}
```

OrderSummaryViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class OrderSummaryViewModel
    {
        public ShoppingCart ShoppingCart { get; set; }
        public decimal ShoppingCartTotal { get; set; }
        public Order Order { get; set; }
    }
```

```
}
```

ShoppingViewCartModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class ShoppingCartViewModel
    {
        public ShoppingCart ShoppingCart { get; set; }
        public decimal ShoppingCartTotal { get; set; }
    }
}
```

SlotMachineViewModel.cs

```
using GamesPlatform.Models;

namespace GamesPlatform.ViewModels
{
    public class SlotMachineViewModel
    {
        public string[] Reels { get; set; }
        public bool? IsWinner { get; set; }
        public decimal AmountWon { get; set; }
        public CasinoResult CasinoResult { get; set; }
    }
}
```

WalletViewModel.cs

```
namespace GamesPlatform.ViewModels
{
    public class WalletViewModel
    {
        public decimal Balance { get; set; }
    }
```

```
}
```

Views

CreateRole.cshtml

```
@model CreateRoleViewModel

@{
    ViewBag.Title = "Create New Role";
}
<h1 class="text-white">Create Role</h1>

<form asp-action="CreateRole" method="post" class="mt-3">
    <div asp-validation-summary="All" class="text-danger">
    </div>
    <div class="form-group row">
        <label asp-for="RoleName" class="col-sm-2 col-form-label"></label>
        <div class="col-sm-10">
            <input asp-for="RoleName" class="form-control"
placeholder="Name">
            <span asp-validation-for="RoleName"
class="text-danger"></span>
        </div>
    </div>

    <div class="form-group row">
        <div class="col-sm-10">
            <button type="submit" class="btn btn-primary"
style="width:auto">
                Create Role
            </button>
        </div>
    </div>
</form>
```

Index.cshtml

```
<h1 class="text-white">Welcome to the Admin Page</h1>
<h3 class="text-white">Select Verification Requests in order to
approve/decline pending verification applications.</h3>

<a asp-action="ViewVerificationRequests">
    <input class="btn btn-primary text-white" type="button"
value="Verification Requests" />
</a>
```

UpdateVerificationRequest.cshtml

```
@model Verification

@{
    ViewData["Title"] = "Update Verification Request";
}

<h4 class="text-white">Update Verification Requests</h4>
<p class="text-white">Accept images that have followed proper guidelines
on Verification submission and of legal age in related country.</p>
<hr />
<div class="row">
    <div class="col-md-6">
        <form asp-action="UpdateRequestPost" class="card">
            <div class="card-body text-dark">
                <table class="table table-dark">
                    <tbody>
                        <tr>
                            <td><label asp-for="VerificationID"
class="control-label">Verification ID:</label></td>
                            <td><strong>@Html.DisplayFor(modelItem =>
Model.VerificationID)</strong></td>
                        </tr>
                        <tr>
                            <td><label asp-for="UserID"
class="control-label">User ID:</label></td>
                            <td><strong>@Html.DisplayFor(modelItem =>
Model.UserID)</strong></td>
                        </tr>
```

```
                        <tr>
                            <td><label asp-for="DateOfRequest"
class="control-label">Date of Request:</label></td>
                            <td><strong>@Html.DisplayFor(modelItem =>
Model.DateOfRequest)</strong></td>
                        </tr>
                        <tr>
                            <td><label asp-for="Status"
class="control-label">Current Status:</label></td>
                            <td><strong>@Html.DisplayFor(modelItem =>
Model.Status)</strong></td>
                        </tr>
                        <tr>
                            <td><label asp-for="Status"
class="control-label">Update Status:</label></td>
                            <td>
                                <select asp-for="Status"
class="form-control">
                                    <option
value="Approved">Approved</option>
                                    <option value="Denied">Denied</option>
                                </select>
                                <span asp-validation-for="Status"
class="text-danger"></span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            <div class="form-group text-dark d-flex">
                <input type="submit" value="Update" class="btn
btn-primary" asp-route-verificationId="@Model.VerificationID" />
                <a class="btn btn-secondary ml-2"
asp-action="ViewVerificationRequests">Back to List</a>
            </div>
            </div>
        </form>
    </div>

    <div class="col-md-4">
        <div class="card">
```

```
            <div class="card-header text-white">
                Photo ID
            </div>
            <img src="@ViewData["Image"]"
height="445" width="300" class="card-img-bottom" />

        </div>
    </div>
</div>
```

ViewVerificationrequest.cshtml

```
@model PaginatedList<Verification>

    @{
        ViewData["Title"] = "View Verification Requests";
    }
    <p>
        <a asp-action="Index"></a>
    </p>
    <h2 class="text-white">View All Verification Requests</h2>
    <h3 class="text-white">Approve or decline verification requests for
Users 18+</h3>

    <form asp-action="ViewVerificationRequests" method="get">
        <div class="form-actions no-color">
    <p class="text-white d-inline-block">
        Search by user ID: <input type="text" name="SearchString"
value="@ViewData["CurrentFilter"]" />
        <input type="submit" value="Search" class="btn btn-primary" />
    </p>
    <a class="btn btn-secondary d-inline-block ml-2"
asp-action="ViewVerificationRequests">Refresh</a>
        </div>
    </form>

    <table class="table table-dark table-striped table-hover">
        <thead>
            <tr>
                <th class ="text-white">
```

```
                    <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["StatusSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Status</a>
                </th>
                <th>
                    User ID
                </th>
                <th class="text-white">
                    <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["DateOfRequestSortParam"]"
asp-route-currentFilter="@ViewData["CurrentFilter"]">Date Of Request</a>
                </th>
                <th>

                </th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
                    <td>
                        @Html.DisplayFor(modelItem => item.Status)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.UserID)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.DateOfRequest)
                    </td>
                    <td>
                        <a asp-action="UpdateVerificationRequest"
asp-route-verificationId="@item.VerificationID">Update</a>
                    </td>
                </tr>
            }
        </tbody>
    </table>

    @{
```

```
        var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
        var nextDisabled = !Model.HasNextPage ? "disabled" : "";
    }

    <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex - 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @prevDisabled">Previous</a>
    <a asp-action="ViewVerificationRequests"
asp-route-sortOrder="@ViewData["CurrentSort"]"
asp-route-pageNumber="@(Model.PageIndex + 1)"
asp-route-currentFilter="@ViewData["CurrentFilter"]" class="btn
btn-primary @nextDisabled">Next</a>
```

BlackJack
Index.cshtml

```
@model BlackJackViewModel
@{
    ViewData["Title"] = "Blackjack";
}

<h1 class="text-white">@ViewData["Title"]</h1>

<form asp-action="Play" method="post">
    <button type="submit" class="btn btn-primary">Play</button>
</form>

@if (Model?.IsWinner != null)
{
        <h2 class="text-white">Player Hand: @string.Join(", ",
Model.PlayerHand) - Score: @Model.PlayerHand.Sum()</h2>
        <h2 class="text-white">Dealer Hand: @string.Join(", ",
Model.DealerHand) - Score: @Model.DealerHand.Sum()</h2>
        <h2 class="@(Model.IsWinner.Value ? "text-success" :
"text-danger")">@(Model.IsWinner.Value ? "You won!" : "You lost.")</h2>
    <h3 class="text-white">Game Details:</h3>
    <ul class="text-white">
```

```
            <li class="text-white">User Choice:
@Model.CasinoResult.UserChoice</li>
        <li class="text-white">Computer Choice:
@Model.CasinoResult.ComputerChoice</li>
        <li class="text-white">Result: @Model.CasinoResult.Result</li>
        <li class="text-white">Amount Won:
@Model.CasinoResult.AmountWon</li>
        <li class="text-white">Date Result Placed:
@Model.CasinoResult.DateResultPlaced</li>
        </ul>
}
```

Casino
Index.cshtml

```
@model CasinoViewModel
@{
    ViewData["Title"] = "Casino";
}
@if (ViewBag.VerificationMessage != null)
{
    <div class="alert alert-info">
        @ViewBag.VerificationMessage
    </div>
}

@if (ViewBag.VerificationMessage == null ||
!User.Identity.IsAuthenticated)
{
    <h2 class="text-white">Casino Games</h2>

    <div class="row">
        @foreach (var game in Model.CasinoGame)
        {
            @Html.Partial("GameCard", game)
        }
    </div>
}
```

CoinFlip
Index.cshtml

```
@model CoinFlipViewModel

@{
    ViewData["Title"] = "Coin Flip";
}

<h1 class="text-white">@ViewData["Title"]</h1>

<div class="cf-container">
    <div class="cf-coin" id="coin">
        <div class="cf-heads">H</div>
        <div class="cf-tails">T</div>
    </div>

    <div class="cf-controls">
        <form asp-action="Flip" method="post" onsubmit="return
flipCoinAnimation(event)">
            <div class="form-group">
                <label for="PlayerChoice"
class="text-white"><strong>Choose Heads or Tails:</strong></label>
                <select id="PlayerChoice" name="PlayerChoice"
class="form-control">
                    <option value="Heads">Heads</option>
                    <option value="Tails">Tails</option>
                </select>
            </div>
            <button type="submit" class="btn btn-primary cf-button">Flip
the Coin</button>
        </form>
    </div>
</div>

@if (Model?.IsWinner != null)
{
    <div class="cf-results">
        <h2 class="text-white">Result: @Model.Result</h2>
```

```
        <h2 class="@(Model.IsWinner.Value ? "text-success" :
"text-danger")">@(Model.IsWinner.Value ? "You won!" : "You lost.")</h2>
        <h3 class="text-white">Game Details:</h3>
        <ul class="text-white cf-details-list">
            <li class="text-white">User Choice:
@Model.CasinoResult.UserChoice</li>
            <li class="text-white">Computer Choice:
@Model.CasinoResult.ComputerChoice</li>
            <li class="text-white">Result: @Model.CasinoResult.Result</li>
            <li class="text-white">Amount Won:
@Model.CasinoResult.AmountWon</li>
        </ul>
    </div>
}
<script src="~/js/coin.js" asp-append-version="true"></script>
```

Contact
Index

```
<h1> Contact us </h1>
<p>If you have any questions please do not hesitate to contact us:</p>

<email address="supportGamesino@outlook.com" content="Email us for
Issues!"></email>
```

Error
Error.cshtml

```
<h2 class="text-white" >Error Error Error!!!!! Sound the alarm!</h2>
```

Games
Details.cshtml

```
@model Game
@{
    bool userHasGame = ViewData.ContainsKey("UserHasGame") &&
(bool)ViewData["UserHasGame"];
}

<div class="card col-12 col-md-6 col-lg-4">
```

```
    <div class="card-img-container">
        <img class="card-img-top" src="@Model.GameImageThumbnailUrl"
alt="Card image cap">
    </div>
    <div class="card-bottom text-white">
        <h5 class="card-title">@Model.GameName</h5>
        <p class="card-text">@Model.GameDescription</p>
        <p class="card-text">€@Model.Price</p>
        @if (Model.CategoryID == 3)
        {
            <form method="post" asp-controller="PlayGame"
asp-action="SubtractFromWalletAndPlayGame">
                <input type="hidden" name="gameID" value="@Model.GameID"
/>
                <button type="submit" class="btn btn-success mt-auto">Play
Game for €5</button>
            </form>
            @if (TempData["ErrorMessage"] != null)
            {
                <div class="alert alert-danger mt-2">
                    <p>@TempData["ErrorMessage"]</p>
                    <a class="btn btn-primary" asp-controller="Withdraw"
asp-action="Index">Add Funds</a>
                </div>
            }
        }
        else if (Model.CategoryID >= 4)
        {
            <div class="startFreeGame">
                <a class="btn btn-success mt-auto start-game"
asp-controller="PlayGame" asp-action="PlayGame"
asp-route-gameId="@Model.GameID">Start Game</a>
            </div>
        }
        else
        {
            <div class="addToCart text-right text-white">
                @if (ViewData["UserHasGame"] != null &&
(bool)ViewData["UserHasGame"] == true)
                {
```

```
                    <a class="btn btn-success text-white"
asp-controller="PlayGame" asp-action="PlayGame"
asp-route-gameId="@Model.GameID">Play Game</a>
                }
                else
                {
                    <a class="btn btn-success text-white" id="cartButton"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
asp-route-gameID="@Model.GameID">Add To Cart</a>
                }
            </div>
        }
    </div>
</div>
```

List.cshtml

```
@using GamesPlatform.ViewModels
@model GameListViewModel
@{
    ViewData["Title"] = "List of avaliable games";
}

<h1 class="text-white">@Model.CurrentCategory</h1>

<div class="row">
@{
    foreach (Game game in Model.Games)
    {
        bool userHasGame = Model.PurchasedGameIDs.Contains(game.GameID);
        var partialViewData = new ViewDataDictionary(ViewData)
        {
            { "UserHasGame", userHasGame }
        };
        @Html.Partial("GameCard", game, partialViewData)
    }
}
</div>
```

Home

Error404.cshtml

```
<head>
    <link
href="https://fonts.googleapis.com/css2?family=Nunito+Sans:wght@600;900&di
splay=swap" rel="stylesheet">
    <script src="https://kit.fontawesome.com/4b9ba14b0f.js"
crossorigin="anonymous"></script>
</head>
<body>
    <div class="er-mainbox">
        <div class="er-err text-white">4</div>
        <i class="far fa-question-circle fa-spin"></i>
        <div class="er-err2">4</div>
        <div class="er-msg text-white">Maybe this page moved? Got deleted?
Is hiding out in quarantine? Never existed in the first place?
            <p class="text-white">Let's go back to <a class="text-primary
er-a" asp-area="" asp-controller="Home" asp-action="Index">Home</a> and
try from there.</p>
        </div>
    </div>
</body>
```

Index.cshtml

```
@model HomeViewModel
@{
    ViewData["Title"] = "Home";
}

@await Html.PartialAsync("Carousel")

<h2 class="text-white">Recommended Games</h2>

<div class="row">
    @foreach(var game in Model.FeaturedGames)
    {
        bool userHasGame = Model.PurchasedGameIDs.Contains(game.GameID);
        var partialViewData = new ViewDataDictionary(ViewData)
        {
            { "UserHasGame", userHasGame }
```

```
        };
        @Html.Partial("GameCard", game, partialViewData)
    }
</div>

<h2 class="text-white">Free Browser Games</h2>

<div class="row mb-2">
    @foreach (var game in Model.FreeGames)
    {
        @Html.Partial("GameCard", game)
    }
</div>
```

Privacy.cshtml

```
@{
    ViewData["Title"] = "Privacy Policy";
}

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <h1 class="text-white">@ViewData["Title"]</h1>
</head>
<body class="text-white">
    <div class="container">
        <h1>Privacy Policy for Gamesino</h1>
        <p>
            This Privacy Policy applies to the Gamesino application, a
Final Year Project developed
            by Samuel David at South East Technological University -
Carlow. This policy is not intended for a real-life
            production app and is provided for educational purposes only.
By using the App, you agree to the collection
            and use of information in accordance with this Privacy Policy.
        </p>
```

```
        <h2>1. Information Collection and Use</h2>

        <p>
            While using the App, we may ask you to provide certain
personally identifiable information that can be used
            to contact or identify you ("Personal Data"). Personally
identifiable information may include, but is not
            limited to:
        </p>

        <ul>
            <li>Email address</li>
            <li>First name and last name</li>
            <li>Phone number</li>
            <li>Address, State, Province, ZIP/Postal code, City</li>
        </ul>

        <h2>2. Use of Data</h2>

        <p>Gamesino uses the collected data for various purposes:</p>

        <ul>
            <li>To provide and maintain the App</li>
            <li>To notify you about changes to the App</li>
            <li>To provide customer care and support</li>
            <li>To gather analysis or valuable information to improve the
App</li>
            <li>To monitor the usage of the App</li>
        </ul>

        <h2>3. Disclosure of Data</h2>

        <p>We will not disclose your Personal Data to third parties,
except in the following cases:</p>

        <ul>
            <li>To comply with a legal obligation</li>
            <li>To protect and defend the rights or property of [Your
Project Name]</li>
```

```
            <li>To prevent or investigate possible wrongdoing in
connection with the App</li>
            <li>To protect the personal safety of users of the App or the
public</li>
            <li>To protect against legal liability</li>
        </ul>

        <h2>4. Security of Data</h2>

        <p>
            The security of your data is important to us, but remember
that no method of transmission over the Internet,
            or method of electronic storage is 100% secure. While we
strive to use commercially acceptable means to
            protect your Personal Data, we cannot guarantee its absolute
security.
        </p>

        <h2>5. Changes to This Privacy Policy</h2>

        <p>
            We may update our Privacy Policy from time to time. We will
notify you of any changes by posting the new
            Privacy Policy on this page. You are advised to review this
Privacy Policy periodically for any changes.
            Changes to this Privacy Policy are effective when they are
posted on this page.
        </p>

        <h2>6. Contact Us</h2>

        <p>If you have any questions about this Privacy Policy, please
contact us by email: C00250105@itcarlow.ie</p>

        <p>Last updated: 15/04/2023</p>
    </div>
    </body>
```

Library
Index.cshtml

```
@model LibraryViewModel
@{
    ViewData["Title"] = "My GameLibrary";
}

@if (Model.PurchasedGames.Count > 0)
{
    <h2 class="text-white">These are the current Games in your
library</h2>
    <div class="row">
        @foreach (var game in Model.PurchasedGames)
        {
            var partialViewData = new ViewDataDictionary(ViewData)
        {
            { "UserHasGame", true }
        };
            @Html.Partial("GameCard", game, partialViewData)
        }
    </div>

}
else {
    <h2 class="text-white">You currently have no games in your library.
Grab your first game from the store</h2>
}
```

Order
Checkout.cshtml

```
@model Order

<section class="vh-100 gradient-custom">
    <div class="container py-5 h-100">
        <div class="row d-flex justify-content-center align-items-center
h-100">
```

```
            <div class="col-12 col-md-8 col-lg-6 col-xl-5">
                <div class="card bg-dark text-white" style="border-radius:
1rem;">
                    <div class="card-body p-5">
                        <h3 class="text-center text-white mb-5">You are
just one step away from ordering your Games.</h3>
                        <form asp-action="Checkout" method="post"
class="form-horizontal text-white" role="form">
                            <div class="row">
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="FirstName"
class="form-label"></label>
                                        <input asp-for="FirstName"
class="form-control" />
                                        <span
asp-validation-for="FirstName" class="text-danger"></span>
                                    </div>
                                </div>
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="LastName"
class="form-label"></label>
                                        <input asp-for="LastName"
class="form-control" />
                                        <span
asp-validation-for="LastName" class="text-danger"></span>
                                    </div>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="AddressLine1"
class="form-label"></label>
                                        <input asp-for="AddressLine1"
class="form-control" />
                                        <span
asp-validation-for="AddressLine1" class="text-danger"></span>
```

```
                                    </div>
                                </div>
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="AddressLine2"
class="form-label"></label>
                                        <input asp-for="AddressLine2"
class="form-control" />
                                        <span
asp-validation-for="AddressLine2" class="text-danger"></span>
                                    </div>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="ZipCode"
class="form-label"></label>
                                        <input asp-for="ZipCode"
class="form-control" />
                                        <span asp-validation-for="ZipCode"
class="text-danger"></span>
                                    </div>
                                </div>
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="City"
class="form-label"></label>
                                        <input asp-for="City"
class="form-control" />
                                        <span asp-validation-for="City"
class="text-danger"></span>
                                    </div>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
```

```
                                        <label asp-for="Country"
class="form-label"></label>
                                        <input asp-for="Country"
class="form-control" />
                                        <span asp-validation-for="Country"
class="text-danger"></span>
                                    </div>
                                </div>
                                <div class="col-md-6 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="PhoneNumber"
class="form-label"></label>
                                        <input asp-for="PhoneNumber"
class="form-control" />
                                        <span
asp-validation-for="PhoneNumber" class="text-danger"></span>
                                    </div>
                                </div>
                            </div>

                            <div class="row">
                                <div class="col-md-12 mb-4">
                                    <div class="form-outline">
                                        <label asp-for="Email"
class="form-label"></label>
                                        <input asp-for="Email"
class="form-control" />
                                        <span asp-validation-for="Email"
class="text-danger"></span>
                                    </div>
                                </div>
                            </div>

                            <div class="form-group mt-3">
                                <div class="d-flex
justify-content-center">
                                    <button class="btn btn-success"
type="submit" asp-controller="Payments" asp-action="Index">Review
Order</button>
```

```
                                    @*<input class="btn btn-default"
type="submit" value="Return " />*@
                                </div>
                            </div>

                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```

CheckoutComplete.cshtml

```
<h1 class="text-white">
    @ViewBag.CheckoutCompleteMessage
</h1>
```

Index.cshtml

```
@model List<Order>

    @{
        ViewData["Title"] = "All orders";
    }

    <div class="row text-white">
        <div class="col-md-8 offset-2">
            <p>
                <h4 class="text-white">List of all your orders</h4>
            </p>

            <table class="table  table-striped bg-dark text-white"
style="border-radius: 1rem;"">
                <thead>
                    <tr>
                        <th>Order ID</th>
                        <th>Items</th>
                        <th>Total</th>
                        <th>User</th>
```

```
                    </tr>
                </thead>
                <tbody>
                    @foreach (var order in Model)
                    {
                        <tr >
                            <td class="align-middle
text-white">@order.OrderID</td>
                            <td class="align-middle text-white">
                                <ul style="list-style-type:none">
                                    @foreach (var item in
order.OrderDetails)
                                    {
                                        <li>
                                            <div class="alert alert-info"
role="alert">
                                                <span class="badge
bg-primary">@item.Amount</span> [@item.Price.ToString("c")] -
@item.Game.GameName
                                            </div>
                                        </li>
                                    }
                                </ul>
                            </td>
                            <td class="align-middle text-white">
                                @order.OrderDetails.Select(m =>
m.Game.Price * m.Amount).Sum().ToString("c")
                            </td>
                            <td class="align-middle text-white">
@order.UserID </td>
                        </tr>
                    }
                </tbody>
            </table>

        </div>
    </div>
```

Payments

AddFundsSuccess.cshtml

```
<div class="row">
    <div class="col-md-6 offset-3 alert-success text-center">
        <h2>Funds Added Successfully!</h2>
        <p>Your payment was successful, and the funds have been added to
your wallet.</p>
        <hr />
        <p>Thank You King!</p>
        </div>

     <div class="col-md-6 offset-3 text-center">
       <a class="btn btn-primary" asp-controller="Library"
asp-action="Index">Go to Library</a>
     </div>

</div>
```

Cancel.cshtml

```
<head>
    <title>Checkout canceled</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <section>
      <h1 class="text-white">Forgot to add something to your cart? Shop
around then come back to pay!</h1>
    </section>
  </body>
```

Index.cshtml

```
@model OrderSummaryViewModel

@{
    ViewData["Title"] = "Order Summary";
}
<h1 class="text-white">Order Summary</h1>
```

```
<table class="table table-striped bg-dark text-white"
style="border-radius: 1rem;">
    <thead>
        <tr>
            <th class="text-white">Selected Games</th>
            <th class="text-white">Price</th>
            <th class="text-right text-white">Quantity</th>
            <th class="text-right text-white">Sub Total</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.ShoppingCart.ShoppingCartItems)
        {
                <tr>
                <td class="text-left text-white">@item.Game.GameName</td>
                <td class="text-right text-white">€@item.Game.Price</td>
                <td class="text-center text-white">@item.Amount</td>
                <td class="text-right text-white">
                        €@((item.Amount * item.Game.Price).ToString("F"))
                    </td>
                </tr>
        }
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3" class="text-right text-white">Total</td>
            <td class="text-right
text-white">€@((Model.ShoppingCartTotal).ToString("F"))</td>
        </tr>
    </tfoot>
</table>

<div class="text-center text-white">
    <form action="/create-checkout-session" method="POST">
        <button class="btn btn-primary" type="submit">Proceed to
Payment</button>
    </form>
</div>
```

Success.cshtml

```
<div class="row">
    <div class="col-md-6 offset-3 alert-success text-center">
        <h2>Order completed successfully!</h2>
        <p>You can check all your orders in the Transaction history of
your profile</p>
        <hr />
        <p>Thank You King!</p>
        </div>

     <div class="col-md-6 offset-3 text-center">
       <a class="btn btn-primary" asp-controller="Library"
asp-action="Index">Go to Library</a>
     </div>

</div>
```

PlayGame
PlayGame.cshtml

```
@{
    ViewData["Title"] = "Play Game";
}

<h2 class="text-white">Play Game</h2>
<div id="gameContainer"></div>

@section Scripts {
    <script>
        $(document).ready(function () {
            var gameID = '@ViewBag.GameID';
            loadGame(gameID);
        });

        function loadGame(gameID) {
            var gameEmbedCode = '';

            // Replace the following with actual gameID and embed code
mapping
            switch (gameID) {
```

```
                case '2':
                    gameEmbedCode = '<div style="position:
relative;height: 0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/keep-it-up-football-kick-ups-"
scrolling="no" seamless="seamless" frameBorder="0" style="position:
absolute;top:0;left: 0;width: 100%;height: 100%;">Browser not
compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '3':
                gameEmbedCode = '<div style="position: relative;height:
0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/amoeba-online" scrolling="no"
seamless="seamless" frameBorder="0" style="position: absolute;top:0;left:
0;width: 100%;height: 100%;">Browser not compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '4':
                gameEmbedCode = '<div style="position: relative;height:
0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/reaction" scrolling="no" seamless="seamless"
frameBorder="0" style="position: absolute;top:0;left: 0;width:
100%;height: 100%;">Browser not compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '5':
                gameEmbedCode = '<div style="position: relative;height:
0;overflow: hidden;padding-bottom: 56.25%;">' +
```

```
                        '<iframe id="embededGame"
src="https://idev.games/embed/dungeon-deathmatch" scrolling="no"
seamless="seamless" frameBorder="0" style="position: absolute;top:0;left:
0;width: 100%;height: 100%;">Browser not compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '6':
                    gameEmbedCode = '<div style="position:
relative;height: 0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/dave-wakes-up" scrolling="no"
seamless="seamless" frameBorder="0" style="position: absolute;top:0;left:
0;width: 100%;height: 100%;">Browser not compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '7':
                    gameEmbedCode = '<div style="position:
relative;height: 0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/tower-of-hanoi" scrolling="no"
seamless="seamless" frameBorder="0" style="position: absolute;top:0;left:
0;width: 100%;height: 100%;">Browser not compatible.</iframe>' +
                        '</div>'+ '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '8':
                    gameEmbedCode = '<div style="position:
relative;height: 0;overflow: hidden;padding-bottom: 56.25%;">' +
                        '<iframe id="embededGame"
src="https://idev.games/embed/high-or-low---number-guessing-game"
scrolling="no" seamless="seamless" frameBorder="0" style="position:
```

```
absolute;top:0;left: 0;width: 100%;height: 100%;">Browser not
compatible.</iframe>' +
                        '</div>' + '<button
onclick="goFullscreen(\'embededGame\');  return false" style="
background-color: #4CAF50; color: white;  text-align: center;
border-radius: 25px;">Fullscreen</button>';
                break;
                case '9':
                    window.location.href = '@Url.Action("RPSIndex",
"RPS")';
                break;
                case '10':
                    window.location.href = '@Url.Action("Index",
"CoinFlip")';
                break;
                case '12':
                    window.location.href = '@Url.Action("Index",
"SlotMachine")';
                break;
                case '13':
                    window.location.href = '@Url.Action("Index",
"BlackJack")';
                break;
            // Add more cases for other games
            default:
                gameEmbedCode = 'Invalid gameID or game not purchased.';
        }
            $('#gameContainer').html(gameEmbedCode);
        }
        function goFullscreen(iframeId) {
            var iframe = document.getElementById(iframeId);
            if (iframe.requestFullscreen) {
                iframe.requestFullscreen();
            } else if (iframe.mozRequestFullScreen) { /* Firefox */
                iframe.mozRequestFullScreen();
            } else if (iframe.webkitRequestFullscreen) { /* Chrome, Safari
and Opera */
                iframe.webkitRequestFullscreen();
            } else if (iframe.msRequestFullscreen) { /* IE/Edge */
                iframe.msRequestFullscreen();
```

```
            }
        }
    </script>
    }
```

RPS

RPSIndex.cshtml

```
@model CasinoResult

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Rock, Paper, Scissors</title>
    <style>
        /* Add your styles here */
        .win {
            color: green;
        }

        .lose {
            color: red;
        }

        .tie {
            color: orange;
        }
    </style>
</head>
<body class="rps">
    <h1 class="text-white text-center">Rock, Paper, Scissors</h1>
    <div class="game-container">

        <img src="~/image/gameimages/rockk.png" alt="Rock" id="rock"
style="cursor: pointer;" />
        <img src="~/image/gameimages/papers.png" alt="Paper" id="paper"
style="cursor: pointer;" />
        <img src="~/image/gameimages/scissorss.png" alt="Scissors"
id="scissors" style="cursor: pointer;" />
```

```
        @if (Model != null && !string.IsNullOrEmpty(Model.UserChoice))
        {
            var resultClass = Model.Win ? "win" : (Model.Result == "It's a
tie!" ? "tie" : "lose");
            <div>
                <h2 class="text-white">Your choice: @Model.UserChoice</h2>
                <h2 class="text-white">Computer's choice:
@Model.ComputerChoice</h2>
                <h2 class="@resultClass">Result: @Model.Result</h2>
                @if (Model.Win)
                {
                    <h3 class="win text-white">Amount Won:
€@Model.AmountWon</h3>
                }
            </div>
        }
    </div>

    <script src="~/js/site.js" asp-append-version="true"></script>
</body>
</html>
```

Shared

Components
CategoryMenu
Default.cshtml

```
<li class="dropdown">
    <a asp-controller="Games"
       asp-action="List"
       class="dropdown-toggle" data-toggle="dropdown">Games<b
class="caret"></b></a>
    <ul class="dropdown-menu">
        @foreach (var category in Model)
        {
```

```
            <li>
                <a asp-controller="Games" asp-action="List"

asp-route-category="@category.CategoryName">@category.CategoryName</a>
            </li>
        }
        <li class="divider"></li>
        <li>
            <a asp-controller="Games" asp-action="List"
asp-route-category="">View all games</a>
        </li>
    </ul>
</li>
```

ShoppingCartSummary

Default.cshtml

```
@model ShoppingCartViewModel

@if (Model.ShoppingCart.ShoppingCartItems.Count > 0)
{
    <li>
        <a asp-controller="ShoppingCart">
            <span class="glyphicon glyphicon-shopping-cart">
                <svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-cart4" viewBox="0 0 16 16">
                <path d="M0 2.5A.5.5 0 0 1 .5 2H2a.5.5 0 0 1 .485.379L2.89
4H14.5a.5.5 0 0 1 .485.621l-1.5 6A.5.5 0 0 1 13 11H4a.5.5 0 0
1-.485-.379L1.61 3H.5a.5.5 0 0 1-.5-.5zM3.14 5l.5 2H5V5H3.14zM6
5v2h2V5H6zm3 0v2h2V5H9zm3 0v2h1.36l.5-2H12zm1.11 3H12v2h.61l.5-2zM11
8H9v2h2V8zM8 8H6v2h2V8zM5 8H3.89l.5 2H5V8zm0 5a1 1 0 1 0 0 2 1 1 0 0 0
0-2zm-2 1a2 2 0 1 1 4 0 2 2 0 0 1-4 0zm9-1a1 1 0 1 0 0 2 1 1 0 0 0 0-2zm-2
1a2 2 0 1 1 4 0 2 2 0 0 1-4 0z"/>
                </svg>
            </span>
            <span id="cart-status">
                @Model.ShoppingCart.ShoppingCartItems.Count
            </span>
        </a>
```

```
    </li>
}
```

WalletBalance
Default.cshtml

```
@model WalletViewModel

@if (Model.Balance > 0.00M)
{
    <li class="nav-item">
        <a id="winnings" class="nav-link text-white"
asp-controller="Withdraw" asp-action="Index"> Balance: €@Model.Balance</a>
    </li>
}
```

CategoryMenu.cs

```
using GamesPlatform.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace GamesPlatform.Views.Shared.Components
{
    public class CategoryMenu : ViewComponent
    {
        private readonly ICategoryRepository _categoryRepository;
        public CategoryMenu(ICategoryRepository categoryRepository)
        {
            _categoryRepository = categoryRepository;
        }

        public IViewComponentResult Invoke()
        {
            var categories = _categoryRepository.Categories.OrderBy(p =>
p.CategoryName);
            return View(categories);
        }
    }
}
```

_GameCover.cshtml

```
<div class="card" style="width: 18rem;">
    <img class="card-img-top" src="~/image/gameimages/football.jpg"
alt="Card image cap">
    <div class="card-body">
        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some quick example text to build on the card
title and make up the bulk of the card's content.</p>
        <a href="~/Views/Play/PlayGame.cshtml" class="btn
btn-primary">Play</a>
    </div>
</div>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>@ViewData["Title"] - Gamesino</title>
    <link rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true"
/>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm
navbar-light bg-dark text-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand text-white" asp-area=""
asp-controller="Home" asp-action="Index"><img
src="~/image/Logos/GSLogo.png" alt="Gamesino Logo" class="navbar-logo"
height="37"></a>
```

```
                <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle
navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex
justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-area=""
asp-controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white">@await
Component.InvokeAsync("CategoryMenu")</a>
                        </li>
                        <li class="nav-item">
                            @if (User.Identity.IsAuthenticated)
                            {
                                <a class="nav-link text-white" asp-area=""
asp-controller="Library" asp-action="Index">Library</a>
                            }
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white" asp-area=""
asp-controller="Casino" asp-action="Index">Casino</a>
                        </li>

                        <li class="nav-item">
                            @if (User.Identity.IsAuthenticated)
                            {
                                <a class="nav-link text-white" asp-area=""
asp-controller="Order" asp-action="Index">Order History</a>
                            }
                        </li>
                         <li class="nav-item">
                            @if (User.Identity.IsAuthenticated)
                            {
```

```
                                <a class="nav-link text-white" asp-area=""
asp-controller="Withdraw" asp-action="Index">Add Funds</a>
                            }
                        <li class="nav-item">
                            @if (User.Identity.IsAuthenticated &&
User.Identity.Name == "admin@gamesino.com")
                            {
                                <a class="nav-link text-white" asp-area=""
asp-controller="Admin" asp-action="Index">Admin</a>
                            }
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white">@await
Component.InvokeAsync("WalletBalance")</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-white">@await
Component.InvokeAsync("ShoppingCartSummary")</a>
                        </li>
                        <form class="navbar-form navbar-right"
asp-controller="Games" asp-action="List" method="get">
                            <div class="form-group">
                                <input type="text" class="form-control"
placeholder="Search for a Game" name="searchString"
value="@ViewData["searchString"]">
                            </div>
                        </form>
                    </ul>
                    <partial name="_LoginPartial" />
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="text-white border-top footer">
```

```
        <div class="container">
            &copy; 2022 - GamesPlatform - <a asp-area=""
asp-controller="Home" asp-action="Privacy">Privacy</a>
            Made By Samuel David For 4th Year Final Project
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script
src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

_LoginPartial.cshtml

```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<IdentityUser> SignInManager
@inject UserManager<IdentityUser> UserManager

<ul class="navbar-nav">
    @if (SignInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a class="nav-link text-white" asp-area="Identity"
asp-page="/Account/Manage/Index" title="Manage">Hello
@User.Identity?.Name!</a>
        </li>
        <li class="nav-item">
            <form class="form-inline" asp-area="Identity"
asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index",
"Home", new { area = "" })">
                <button type="submit" class="nav-link btn btn-link
text-white">Logout</button>
            </form>
        </li>
    }
    else
    {
```

```
        <li class="nav-item nav-item-dropdown dropdown">
            <a class="nav-link text-white dropdown-toggle" href="#"
id="navbarDropdown" role="button" data-bs-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                <svg xmlns="http://www.w3.org/2000/svg" width="30"
height="25" fill="currentColor" class="bi bi-person-circle" viewBox="0 0
16 16">
                 <path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>
                  <path fill-rule="evenodd" d="M0 8a8 8 0 1 1 16 0A8 8 0 0
1 0 8zm8-7a7 7 0 0 0-5.468 11.37C3.242 11.226 4.805 10 8 10s4.757 1.225
5.468 2.37A7 7 0 0 0 8 1z"/>
                </svg>
                 Log In
            </a>
            <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                <a class="dropdown-item" asp-area="Identity"
asp-page="/Account/Register">Register</a>
                <a class="dropdown-item" asp-area="Identity"
asp-page="/Account/Login">Login</a>
            </div>
        </li>
    }
</ul>
```

_ValidationScriptsPartial.cshtml

```
<script
src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script
src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.j
s"></script>
```

Carousel.cshtml

```
<div id="carouselExampleCaptions" class="carousel slide"
data-bs-ride="false">
    <div class="carousel-indicators">
      <button type="button" data-bs-target="#carouselExampleCaptions"
data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide
1"></button>
```

```
      <button type="button" data-bs-target="#carouselExampleCaptions"
data-bs-slide-to="1" aria-label="Slide 2"></button>
      <button type="button" data-bs-target="#carouselExampleCaptions"
data-bs-slide-to="2" aria-label="Slide 3"></button>
    </div>
    <div class="carousel-inner">
      <div class="carousel-item active">
              <img src="~/image/gameimages/ReactionSpeedTestt.png"
class="d-block w-100" width="696" height="493.16" alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>Reaction Speed Test</h5>
          <p>Test your reaction time and compare with others.</p>
        </div>
      </div>
      <div class="carousel-item">
              <img src="~/image/gameimages/football.jpg" class="d-block
w-100" width="696" height="493.16"  alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>Kick-Ups</h5>
          <p>The goal is simple, keep the ball in the air for the longest
amount of time....</p>
        </div>
      </div>
      <div class="carousel-item">
              <img src="~/image/gameimages/DAVE WAKES UP!.png"
class="d-block w-100" width="696" height="493.16"  alt="...">
        <div class="carousel-caption d-none d-md-block">
          <h5>DAVE WAKES UP!</h5>
          <p>Dave wakes up is a short narrative top down 2d game. You play
as Dave, trapped in a corpo world!.</p>
        </div>
      </div>
    </div>
    <button class="carousel-control-prev" type="button"
data-bs-target="#carouselExampleCaptions" data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button"
data-bs-target="#carouselExampleCaptions" data-bs-slide="next">
```

```
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
    </button>
  </div>
```

Error.cshtml

```
@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your
request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more
detailed information about the error that occurred.
</p>
<p>
    <strong>The Development environment shouldn't be enabled for deployed
applications.</strong>
    It can result in displaying sensitive information from exceptions to
end users.
    For local debugging, enable the <strong>Development</strong>
environment by setting the <strong>ASPNETCORE_ENVIRONMENT</strong>
environment variable to <strong>Development</strong>
    and restarting the app.
</p>
```

GameCard.cshtml

```
@model Game
@{
    bool userHasGame = ViewData.ContainsKey("UserHasGame") &&
(bool)ViewData["UserHasGame"];
}

<div class="card col-12 col-md-6 col-lg-4">
  <div class="card-img-container">
    <img class="card-img-top" src="@Model.GameImageThumbnailUrl" alt="Card
image cap">
  </div>
  <div class="card-bottom text-white">
    <h5 class="card-title">@Model.GameName</h5>
    <p class="card-text">@Model.GameDescription</p>
    <p class="card-text">€@Model.Price</p>
    @if (Model.CategoryID == 3)
    {
    <form method="post" asp-controller="PlayGame"
asp-action="SubtractFromWalletAndPlayGame">
        <input type="hidden" name="gameID" value="@Model.GameID" />
        <button type="submit" class="btn btn-success mt-auto">Play Game
for €5</button>
    </form>
        @if (TempData["ErrorMessage"] != null)
        {
            <div class="alert alert-danger mt-2">
                <p>@TempData["ErrorMessage"]</p>
                <a class="btn btn-primary" asp-controller="Withdraw"
asp-action="Index">Add Funds</a>
            </div>
        }
    }
    else if (Model.CategoryID >= 4)
    {
        <div class="startFreeGame">
            <a class="btn btn-success mt-auto start-game"
asp-controller="PlayGame" asp-action="PlayGame"
asp-route-gameId="@Model.GameID">Start Game</a>
        </div>
```

```
    }
    else
    {
        <div class="addToCart text-right text-white">
            @if (ViewData["UserHasGame"] != null &&
(bool)ViewData["UserHasGame"] == true)
            {
                <a class="btn btn-success text-white"
asp-controller="PlayGame" asp-action="PlayGame"
asp-route-gameId="@Model.GameID">Play Game</a>
            }
            else
            {
                <a class="btn btn-success text-white" id="cartButton"
asp-controller="ShoppingCart" asp-action="AddToShoppingCart"
asp-route-gameID="@Model.GameID">Add To Cart</a>
            }
        </div>
    }
  </div>
</div>
```

ShoppingCart
Index.cshtml

```
@model ShoppingCartViewModel

<h2 class="text-white">Shopping Cart:</h2>
<table class="table  table-striped bg-dark text-white"
style="border-radius: 1rem;">
    <thead class="text-white">
        <tr class="text-white">
            <th> </th>
            <th class="text-white">Quantity</th>
            <th class="text-white">Game</th>
            <th class="text-right text-white">Price</th>
            <th class="text-right text-white">Sub Total</th>
        </tr>
    </thead>
    <tbody>
```

```
        @foreach (var item in Model.ShoppingCart.ShoppingCartItems)
        {
            <tr class="text-white">
                    <td class="text-center "><a class="btn btn-danger"
asp-controller="ShoppingCart" asp-action="RemoveFromShoppingCart"
asp-route-gameID="@item.Game.GameID">Remove From Cart</a></td>
                <td class="text-center text-white">@item.Amount</td>
                <td class="text-left text-white">@item.Game.GameName</td>
                <td class="text-right
text-white">€@item.Game.Price.ToString("F")</td>
                <td class="text-right text-white">€@((item.Amount *
item.Game.Price).ToString("F"))
                </td>
            </tr>
        }
    </tbody>
    <tfoot>
        <tr>
            <td colspan="4" class="text-right text-white">Total</td>
            <td class="text-right
text-white">€@Model.ShoppingCartTotal.ToString("F")</td>
        </tr>
    </tfoot>
</table>

<div class="text-center text-white">
 <a class="btn btn-success @(ViewBag.IsEmpty ? "disabled" : "")"
asp-controller="Order" asp-action="Checkout">Check out</a>
    <a class="btn btn-primary" asp-controller="Home"
asp-action="Index">Continue Shopping</a>
    <a class="btn btn-danger" asp-controller="ShoppingCart"
asp-action="ClearCart">Clear Cart</a>
</div>
```

SlotMachine
Index.cshtml

```
@model SlotMachineViewModel
@{
    ViewData["Title"] = "Slot Machine";
```

```
}

<h1 class="text-white">@ViewData["Title"]</h1>

<form asp-action="Spin" method="post">
    <button type="submit" class="btn btn-primary">Spin</button>
</form>

@if (Model?.IsWinner != null)
{
    <h2 class="text-white">Reels: @string.Join(" | ", Model.Reels)</h2>
        <h2 class="@(Model.IsWinner.Value ? "text-success" :
"text-danger")">@(Model.IsWinner.Value ? "You won!" : "You lost.")</h2>
    <h3 class="text-white">Game Details:</h3>
    <ul class="text-white">
            <li class="text-white">User Choice:
@Model.CasinoResult.UserChoice</li>
        <li class="text-white">Computer Choice:
@Model.CasinoResult.ComputerChoice</li>
        <li class="text-white">Result: @Model.CasinoResult.Result</li>
        <li class="text-white">Amount Won:
@Model.CasinoResult.AmountWon</li>
        <li class="text-white">Date Result Placed:
@Model.CasinoResult.DateResultPlaced</li>
        </ul>
}
```

StripePayment

Index

```
@model Stripe.Checkout.Session
@{
    ViewBag.Title = "Home Page";
}
```

Verification

Index.cshtml

```
@model FileUpload
@if (ViewBag.SuccessMessage != null)
{
        <p class="text-success">@ViewBag.SuccessMessage</p>
}
@if (ViewBag.ErrorMessage != null)
{
        <p class="text-danger">@ViewBag.ErrorMessage</p>
}
<h1 class="text-white">Verify your account</h1>
<p class="text-white">To play casino games, you must verify your photo ID
to prove you are genuine and you over the age of 18. This is not required
to play and purchase video games</p>

<h3 class="text-white">Max file size is 2MB. Accepted file types are .jpg
or .png</h3>
<form asp-action="Index" enctype="multipart/form-data" method="post">
    <div class="form-group">
        <label asp-for="file"></label>
        <input asp-for="file" class="form-control" type="file"/>
        <span asp-validation-for="file" class="text-danger"></span>
    </div>
    <input type="submit" class="btn btn-primary" value="Submit" />
    @*<button type="submit" class="btn btn-primary" value="Complete Order"
asp-controller="Payment" asp-action="Index">Proceed</button>*@
</form>
```

Withdraw
Index.cshtml

```
@model Withdraw
@{
    ViewData["Title"] = "Transfer Funds";
}

<h1 class="text-white">Transfer winnings to Paypal</h1>

<h3 class="text-white">If you have a PayPal account, enter your registered
PayPal email and the amount you want to transfer and click 'Transfer
Funds'</h3>
```

```
<form asp-action="Index" method="post" class="card"
style="background-color: #2b2b2b; color: white;">
    <div class="card-body" style="background-color: #2b2b2b; color:
white;">

        <h4 class="card-title">Transfer Balance to PayPal</h4>

        <div class="form-group">
            <label asp-for="PayPalEmail"></label>
            <input asp-for="PayPalEmail" class="form-control" />
            <span asp-validation-for="PayPalEmail"
class="text-danger"></span>
        </div>

        <div class="form-group">
            <label asp-for="AmountTransfered"></label>
            <input asp-for="AmountTransfered" class="form-control" />
            <span asp-validation-for="AmountTransfered"
class="text-danger"></span>
        </div>
        <div class="form-group">
            <input type="submit" class="btn btn-primary" value="Submit" />
        </div>

        @if (ViewBag.SuccessMessage != null)
        {
            <p class="text-success">@ViewBag.SuccessMessage</p>
        }
        @if (ViewBag.ErrorMessage != null)
        {
            <p class="text-danger">@ViewBag.ErrorMessage</p>
        }
    </div>
</form>

            <div class="text-center my-4">
    <hr />
    <h4 class="d-inline mx-3 text-white">Or</h4>
    <hr />
</div>
```

```
<h4 class="text-white">Add Funds to your Wallet</h4>
<div class="row mb-3">
    <div class="col-md-6">
        <div class="card" style="background-color: #2b2b2b; color:
white;">
            <div class="card-body">
                <form asp-action="CreateFundsCheckoutSession"
asp-controller="Payments" method="post">
                    <input type="hidden" name="amount" value="5" />
                    <div class="d-flex align-items-center
justify-content-between">
                        <h5 class="card-title mb-0">Add 5€</h5>
                        <small class="text-muted">(Minimum Fund
Level)</small>
                        <button type="submit" class="btn btn-success">Add
Funds</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<div class="row mb-3">
    <div class="col-md-6">
        <div class="card" style="background-color: #2b2b2b; color:
white;">
            <div class="card-body">
                <form asp-action="CreateFundsCheckoutSession"
asp-controller="Payments" method="post">
                    <input type="hidden" name="amount" value="10" />
                    <div class="d-flex align-items-center
justify-content-between">
                        <h5 class="card-title mb-0">Add 10€</h5>
                        <button type="submit" class="btn btn-success">Add
Funds</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
```

```
</div>
<div class="row mb-3">
    <div class="col-md-6">
        <div class="card" style="background-color: #2b2b2b; color:
white;">
            <div class="card-body">
                <form asp-action="CreateFundsCheckoutSession"
asp-controller="Payments" method="post">
                    <input type="hidden" name="amount" value="20" />
                    <div class="d-flex align-items-center
justify-content-between">
                        <h5 class="card-title mb-0">Add 20€</h5>
                        <button type="submit" class="btn btn-success">Add
Funds</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
```

_ViewImports.cshtml

```
@using GamesPlatform
@using GamesPlatform.Models
@using GamesPlatform.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper GamesPlatform.TagHelpers.*, GamesPlatform
```

_ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

Appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=tcp:mysqldb-user.database.windows.net,1433;Initial
Catalog=userdb;Persist Security Info=False;User
ID=sammerz60322;Password=Oblongata1!;MultipleActiveResultSets=False;Encryp
t=True;TrustServerCertificate=False;Connection Timeout=30;"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "Stripe": {
    "TestSecretKey":
"sk_test_51MLfkELqDZptVo03ItEDQzIaHydvHEVvyIw7SV0Z0GmzqsDWyxV3lEWLkHzfnr4n
rnxxInRobfod8LpmiLdlBqSw00oEP5ziP9",
    "TestPublishableKey":
"pk_test_51MLfkELqDZptVo037xVRGEJbAaHETltBxu5hJjvO9H0doWwJhwvXMJKZm296AmNe
XzSCMdZoFzqYKBEWX3EztAHr00q5ZMQydy"
  },
  "PayPal": {
    "ClientID":
"AVj-IsYoEFbkHc82nnlQncM4R2c9yi47nhnFGLZta8b8ccR3zMiNr7a-fJqq6ZLeEuGhKHCYt
_ZynDC8",
    "Secret":
"EDvu4qdJPqGW6FNc6YpZqYBgv0BiaoKO06ZmtIC2kvrAyuDiP012j52aes3eZhP4XZMeMkPkS
TwG6ajC",
    "UrlAPI": "https://api.sandbox.paypal.com"
  },
  "AllowedHosts": "*"
}
```

PayPalClient.cs

```
using PayoutsSdk.Core;
using PayPalHttp;
using System.Collections.Generic;
```

```
namespace GamesPlatform
{
    public class PayPalClient
    {

        public static PayPalEnvironment environment()
        {
            return new
SandboxEnvironment(System.Environment.GetEnvironmentVariable("PAYPAL_CLIEN
T_ID") != null ?
System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_ID") :
"AVj-IsYoEFbkHc82nnlQncM4R2c9yi47nhnFGLZta8b8ccR3zMiNr7a-fJqq6ZLeEuGhKHCYt
_ZynDC8",
System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET") != null
? System.Environment.GetEnvironmentVariable("PAYPAL_CLIENT_SECRET") :
"EDvu4qdJPqGW6FNc6YpZqYBgv0BiaoKO06ZmtIC2kvrAyuDiP012j52aes3eZhP4XZMeMkPkS
TwG6ajC");
        }
        public static PayPalHttp.HttpClient client()
        {
            return new PayPalHttpClient(environment());
        }
        public static PayPalHttp.HttpClient client(string refreshToken)
        {
            return new PayPalHttpClient(environment(), refreshToken);
        }
    }
}
```

Program.cs

```
using GamesPlatform.Data;
using GamesPlatform.Interfaces;
using GamesPlatform.Mocks;
using GamesPlatform.Models;
using GamesPlatform.Repositories;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Stripe;
```

```
namespace GamesPlatform
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection") ?? throw
new InvalidOperationException("Connection string 'DefaultConnection' not
found.");
            builder.Services.AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(connectionString));
            builder.Services.AddDatabaseDeveloperPageExceptionFilter();

            builder.Services.AddDefaultIdentity<IdentityUser>(options =>
options.SignIn.RequireConfirmedAccount = true).AddRoles<IdentityRole>()
                .AddEntityFrameworkStores<ApplicationDbContext>();
            builder.Services.AddControllersWithViews();
            builder.Services.AddTransient<IGamesRepository,
GameRepository>();
            builder.Services.AddSingleton<IHttpContextAccessor,
HttpContextAccessor>();
            builder.Services.AddScoped(sp => ShoppingCart.GetCart(sp));
            builder.Services.AddTransient<ICategoryRepository,
CategoryRepository>();
            builder.Services.AddTransient<IOrderRepository,
OrderRepository>();
            builder.Services.AddTransient<ICasinoResultRepository,
CasinoResultRepository>();
            builder.Services.AddTransient<IWalletRepository,
WalletRepository>();
            builder.Services.AddTransient<IWithdrawRepository,
WithdrawRepository>();
            builder.Services.AddTransient<IVerificationRepository,
VerificationRepository>();
            builder.Services.AddMvc();
```

```
            builder.Services.AddAuthorization(options =>
            {
                options.AddPolicy("AdminOnly", policy =>
                    policy.RequireAssertion(context =>
                        context.User.Identity.IsAuthenticated &&
context.User.Identity.Name == "admin@gamesino.com"));
            });

            builder.Services.AddMemoryCache();
            builder.Services.AddSession();
            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (app.Environment.IsDevelopment())
            {
                app.UseMigrationsEndPoint();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                // The default HSTS value is 30 days. You may want to
change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
            app.UseStatusCodePagesWithReExecute("/Home/Error404",
"?statusCode={0}");
            app.UseSession();
            app.UseRouting();
            app.UseAuthorization();
            StripeConfiguration.ApiKey =
"sk_test_51MLfkELqDZptVo03ItEDQzIaHydvHEVvyIw7SV0Z0GmzqsDWyxV3lEWLkHzfnr4n
rnxxInRobfod8LpmiLdlBqSw00oEP5ziP9";
            app.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
            app.MapControllerRoute(
                name: "gameDetails",
```

```
                pattern: "{controller=Games}/{action=Details}/{id?}");
            //app.MapControllerRoute(
                //name: "Library",
                //pattern: "{controller=Library}/{action=Index}/{id?}");
            app.MapRazorPages();

            app.Run();
        }
    }
}
```

References

[1]Rick-Anderson (n.d.). *Introduction to Identity on ASP.NET Core*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio [Accessed 17 Apr. 2023].

[2]stripe.com. (n.d.). *Stripe Checkout*. [online] Available at: https://stripe.com/docs/payments/checkout. [Accessed 17 Apr. 2023].

[3]developer.paypal.com. (n.d.). *Payouts Overview*. [online] Available at: https://developer.paypal.com/docs/payouts/. [Accessed 17 Apr. 2023].

[4]ardalis (n.d.). *Overview of ASP.NET Core MVC*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-7.0. [Accessed 17 Apr. 2023].

[5]Rick-Anderson (n.d.). *Introduction to Identity on ASP.NET Core*. [online] learn.microsoft.com. Available at:

https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio. [Accessed 17 Apr. 2023].

[6] https://idev.games/ [Accessed 17 Apr. 2023].

[7] GitHub. (n.d.). *Implementing age verification. Have it so a user can upload a photo o… · KyleHennessy/CentralGamesPlatform@e1a3597*. [online] Available at: https://github.com/KyleHennessy/CentralGamesPlatform/commit/e1a359783730b6153cfea89612f7855f9f165618 [Accessed 17 Apr. 2023].