**Please tell us about your current project, or a recent project in the last 2 years.**

I have been involved in various projects in different teams over the past 2 years. Just recently I had a payment portal project whereby I had to manage old, new and existing clients. In actual, an upgrade on the existing payment portal.

Currently I am involved in our own in-house payment processor which is a fast and easy to use solution

It can process payments if clients have any of the Six major banks in SA. Clients use this system when they want to fund their accounts with credits or money to perform activities which are bets in this case on our platforms

**What type of system was it?**

It’s an online payment solution for customers to make payments into Hollywood’s account from bank to bank

**What business problem did it solve?**

Clients had challenges in transferring funds in their account. Many clients had to provide POP (Proof of Payment) to have funds credited. Those POP’s would have to be checked, some had to go to physical branches to have their accounts topped up

So, this solution takes the hassle out of manual eft payments and allows our clients to be credited funds into their account immediately once we get the outcome of the transaction.

**What was the technology stack?**

Frontend: asp.net,

Back End C#, .Net 5,

ORM: (Entity Framework) Code First

DATABASE: MS SQL

OTHER: Selenium Web driver

**What technical or project challenges did you face?**

Project challenges- Tight Deadlines, Some unclear requirements

Technical challengers – Making changes to existing code, learning the use of external libraries, payment libraries for security. I had to do a lot of plural sight as well to enhance my knowledge on selenium and other aspects.

**What parts of it did you enjoy?**

Some tasks I didn’t have not much knowledge on where to even start or begin but I was challenged to step out of my comfort zone and learn, putting in the extra hours and effort to ensure work gets done. Brainstorming and coming up with brilliant ideas/ game plan on how to get stuff done was exciting which makes the project itself more interesting.

**What parts didn't you enjoy?**

Continuous Mid project Adjustments, wearing differing hats at different times

**LOGIC**

using System;

using System.Collections.Generic;

using System.Linq;

namespace BlackJacke21

{

public class BlackJack21Game

{

public string PlayBlackJack(List<string> cardInput)

{

var result = string.Empty;

var getDealersCardsValue = GetDealerCardsInHandValue();

var playerCardValue = GetSumOfPlayersCardsInHand(cardInput);

var hasAce = DoesPlayersCardListHaveAces(cardInput);

if ((cardInput.Count() >= 5 && playerCardValue < 21) || (playerCardValue < 21 && playerCardValue > getDealersCardsValue) )

{

result = "Player Won";

}

else {

if (playerCardValue > 21 && hasAce)

{

if (hasAce == true)

{

playerCardValue = (playerCardValue - 11) + 1;

}

if (playerCardValue >= getDealersCardsValue)

{

if (playerCardValue <= 21 || playerCardValue == getDealersCardsValue)

{

result = "Player Won";

}

else

{

result = "Dealer Won";

}

}

else

{

result = "Dealer Won";

}

}

else

{

result = "Dealer Won";

}

}

return result;

}

private bool DoesPlayersCardListHaveAces(List<string> cardInput)

{

if (cardInput.Any(str => str.Contains("Ace")))

{

return true;

}

return false;

}

public int GetDealerCardsInHandValue()

{

var cardInput = new List<string>() { "Jack of Spades", "Nine of Hearts" };

return GetSumOfPlayersCardsInHand(cardInput);

}

public int GetSumOfPlayersCardsInHand(List<string> cardInput)

{

var sum = 0;

foreach (var card in cardInput)

{

var cardValue = GetCardValue(card);

sum = sum + cardValue;

}

return sum;

}

private int GetCardValue(string card)

{

var cardName = card.Substring(0, card.IndexOf(" ")).ToLower();

var cardValue = 0;

switch (cardName)

{

case "ace":

cardValue = 11;

break;

case "jack":

cardValue = 10;

break;

case "queen":

cardValue = 10;

break;

case "king":

cardValue = 10;

break;

case "nine":

cardValue = 9;

break;

case "eight":

cardValue = 8;

break;

case "seven":

cardValue = 7;

break;

case "six":

cardValue = 6;

break;

case "five":

cardValue = 5;

break;

case "four":

cardValue = 4;

break;

case "three":

cardValue = 3;

break;

case "two":

cardValue = 2;

break;

case "one":

cardValue = 1;

break;

default:

cardValue = 0;

break;

}

return cardValue;

}

}

}

**TESTS**

using BlackJacke21;

using NUnit.Framework;

using System;

using System.Collections.Generic;

namespace BlackJacke21Tests

{

[TestFixture]

public class BlackJack21GameTests

{

[Test]

public void when\_given\_cards\_return\_sum()

{

// Arrange

var cardInput = new List<string>() { "Jack of Spades", "Nine of Hearts" };

// Act

var sut = new BlackJack21Game();

var result = sut.GetSumOfPlayersCardsInHand(cardInput);

// Assert

Assert.That(result, Is.EqualTo(19));

}

[Test]

public void when\_given\_dealercards\_return\_sum\_of\_DealerCards\_in\_hand()

{

// Arrange

// Act

var sut = new BlackJack21Game();

int result = sut.GetDealerCardsInHandValue();

// Assert

Assert.That(result, Is.EqualTo(19));

}

[Test]

public void when\_player\_places\_seven\_of\_hearts\_seven\_of\_diamonds\_seven\_of\_clubs\_six\_of\_spades\_return\_dealer\_won()

{

// Arrange

var cardInput = new List<string>() { "Seven Of Spades", "Seven of Hearts", "Seven of Diamonds", "Six of Spades" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Dealer Won"));

}

[Test]

public void when\_player\_places\_ace\_of\_hearts\_seven\_of\_diamonds\_seven\_of\_clubs\_six\_of\_spades\_return\_player\_won()

{

// Arrange

var cardInput = new List<string>() { "Ace Of Spades", "Seven of Hearts", "Seven of Diamonds", "Six of Spades" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Player Won"));

}

[Test]

public void when\_player\_places\_ace\_of\_hearts\_seven\_of\_diamonds\_seven\_of\_clubs\_five\_of\_spades\_return\_player\_won()

{

// Arrange

var cardInput = new List<string>() { "Ace Of Spades", "Seven of Hearts", "Seven of Diamonds", "five of Spades" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Player Won"));

}

[Test]

public void when\_player\_places\_an\_ace\_six\_of\_hearts\_ace\_of\_diamonds\_return\_dealer\_won()

{

// Arrange

var cardInput = new List<string>() { "Ace Of Spades", "Six of Hearts", "Ace of Diamonds"};

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Dealer Won"));

}

[Test]

public void when\_player\_places\_an\_ace\_seven\_of\_hearts\_ace\_of\_diamonds\_return\_player\_won()

{

// Arrange

var cardInput = new List<string>() { "Ace Of Spades", "Seven of Hearts", "Ace of Diamonds" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Player Won"));

}

[Test]

public void when\_player\_andrew\_places\_diamonds\_return\_dealer\_won()

{

// Arrange

var cardInput = new List<string>() { "King Of Diamonds", "Four Of Spades", "Four of Clubs" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Dealer Won"));

}

[Test]

public void when\_player\_carl\_places\_queen\_six\_nine\_diamonds\_return\_dealer\_won()

{

// Arrange

var cardInput = new List<string>() { "Queen of Clubs", "Six Of Spades", "Nine of Diamonds" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.That(result, Is.EqualTo("Dealer Won"));

}

[Test]

public void when\_player\_bill\_places\_5\_cards\_and\_less\_than\_twenty\_one\_player\_wins()

{

// Arrange

var cardInput = new List<string>() { "Two of Spades", "Two Of Diamonds", "Two of Hearts", "Four of Diamonds", "Five of Clubs" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.AreEqual("Player Won", result);

}

[Test]

public void when\_player\_cards\_is\_less\_than\_twenty\_one\_but\_greater\_than\_dealer\_cards\_player\_wins()

{

// Arrange

var cardInput = new List<string>() { "King of Spades", "Nine Of Diamonds", "One of Hearts" };

// Act

var sut = new BlackJack21Game();

var result = sut.PlayBlackJack(cardInput);

// Assert

Assert.AreEqual("Player Won", result);

}

}

}