Part-1

Q-2:

1. Provide the name of the Architecture pattern you have just created.

I have use Layered Architecture Pattern to solve using dependency injection.

1. List the benefits of the created Architecture. Elaborate in detail describing the benefits of each module/layer.

Applications that are needed to be built quickly.

Enterprise applications that require traditional IT departments and processes.

Appropriate for teams with inexperienced developers and limited knowledge of architecture patterns.

Applications that require strict standards of maintainability and testability.

1. Which approach have you used to implement the Entity framework within the project?

I used DB first Approach mostly.

1. What security measurement has been covered within the Architecture?

layered security model works by implementing multiple security measures at different layers or levels of a system or network.

Each layer of security provides a unique form of protection, and attackers must bypass multiple layers to gain access to sensitive information.

1. Does your implementation prevent SQL injection? If yes how? Describe in detail

Yes , SQL Injection can prevent because I have used singleton pattern for Unitofwork in DAL layer to access the database.

In this once object of Database will be created in than it will be reuse in current application.

Part-2:

Q-2 :

class Program

{

static void Main()

{

int[,] salesTable = {

{1, 3, 0, 6},

{2, 4, 2, 2},

{3, 6, 1, 1},

{0, 5, 5, 3},

{0, 4, 2, 2},

{0, 2, 1, 1},

{4, 2, 1, 6},

{2, 7, 2, 3},

{0, 1, 3, 1}

};

decimal[] lastYearSales = { 490000, 1000000, 650000 };

decimal[,] commissionRates = {

{800, 0.08m, 0.06m, 0.04m},

{750, 0.06m, 0.05m, 0.03m},

{850, 0.07m, 0.05m, 0.04m},

{400, 0.05m, 0.03m, 0.02m}

};

Console.WriteLine("Salesman\tCommission Amount");

Console.WriteLine("---------------------------------------");

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

{

decimal totalCommission = 0;

for (int j = 0; j < commissionRates.GetLength(0); j++)

{

decimal brandCommission = commissionRates[j, 0];

decimal classACommissionRate = commissionRates[j, 1];

decimal classBCommissionRate = commissionRates[j, 2];

decimal classCCommissionRate = commissionRates[j, 3];

for (int k = 0; k < salesTable.GetLength(0); k++)

{

decimal modelPrice = salesTable[k, j + 1] \* 10000;

decimal classCommission = 0;

if (salesTable[k, 0] == i + 1)

{

if (modelPrice > 25000 && j == 0)

{

classCommission = modelPrice \* classACommissionRate;

}

else if (modelPrice > 35000 && j == 1)

{

classCommission = modelPrice \* classACommissionRate;

}

else if (modelPrice > 30000 && j == 2)

{

classCommission = modelPrice \* classACommissionRate;

}

else if (modelPrice > 20000 && j == 3)

{

classCommission = modelPrice \* classACommissionRate;

}

}

totalCommission += brandCommission + classCommission;

}

}

if (lastYearSales[i] > 500000)

{

totalCommission += 0.02m \* totalCommission;

}

Console.WriteLine($"Salesman {i + 1}\t{totalCommission:C}");

}

}

}