MS.NET

Mini Project

Life Insurance Management System

(LIMS)

Document Revision History

Date	Revision No.	Author	Summary of Changes

Table of Contents

INTRODUCTION

This document outlines a mini project for the .NET Line of Technology (LOT). The project is to develop Life Insurance Management System. This document contains the requirements, work flow of the system and gives guidelines on how to build the functionality gradually in each of the course modules of the .NET LOT.

SETUP CHECKLIST

Minimum System Requirements

- Intel Pentium 4 and above Windows 2007, 2008 and 2010
- Memory 4 GB
- Internet Explorer 8.0 or higher
- SQL Server 2012 client and access to SQL Server 2012 server
- Visual Studio 2017

INSTRUCTIONS

- The code modules in the mini project should follow all the coding standards.
- Create a directory by your name in drive **<drive>**. In this directory, create a subdirectory **MiniProject**. Store your Project here.
- You can refer to your course material.
- You may also look up the help provided in the MSDN
- Since this project work will span over couple of months, you will need to take care of maintaining the code

PROBLEM STATEMENT

OBJECTIVE

Development of Life Insurance Management System (LIMS)

Abstract of the project

Life Insurance Management System (LIMS) helps Administrative Staff of a Life Insurance Company to maintain details of Insurance Plans offered to their customers. There are two actors in this application.

(Note: Currently we have to create application for Administrators only)

1) Administrator: Can perform the following tasks:

- Add Insurance Plan
- Update Insurance Plan
- Delete Insurance Plan
- Can Search for an Insurance Plan and view its details, on the basis of its Plan No.
- Get List of their Insurance Plans and view the details.

2) Customer: Can perform the following tasks:

- Can Search for an Insurance Plan and view its details, on the basis of its Plan No.
- Get List of Insurance Plans and view the details.

- Apply online for an Insurance Plan.

- **Phase 1 :** The system will first develop using C# only – where insurance data will be store as a Collection Classes. For user interaction, system will use Console Application

- **Phase 2 :** Later on data will be store in MS SQL Server database; system will use ADO.NET or LINQ and Entity Framework for the same. User Interface will be designed using WPF

- **Phase 3** : LIMS will become web based application, following MVC design pattern. Here the application will be develop in ASP.NET MVC.

Macro level Operations/offerings:

1. Administrator:

    a. Add Insurance Plan Details: Plan No., Name, Description, Death Benefit, Maturity Benefit, Participation in Profits, etc.

    b. Modify Insurance Plan Details

    c. Remove Insurance Plan

    d. Get List of Insurance Plan and view the details like Plan No., Description, etc.

    e. Search for an Insurance Plan and view its details, on the basis of its Plan No.

2. Customer:

a. Search for an Insurance Plan and view its details, on the basis of its Plan No.

b. Get List of Insurance Plan and view the details like Plan No., Description etc.

c. Apply online for an Insurance Plan.

MODULE LIST and MODULE DETAILS

1) Administrator

CREATE INSURANCE PLAN

Following info need to capture

- PlanNo
- Name
- Description
- Death Benefit
- Maturity Benefit
- Participation In Profits
- Plan Parameters:

Age Of Entry	Example: 18 – 50 Years
Premium Paying Mode	Example:

	Yearly, HalfYearly, Quarterly, Monthly(ECS Only)
Policy Term	Example: 15-35 Years
Basic Sum Assured	Example: 100000 and above
Policy Revival	Example: Within 2 Years
Premium Mode Rebate	Example: 2% on Yearly, 1% on HalfYearly, Nil on Quarterly
Loan	Example: After 3 Years
Surrender	After 3 Years of full premium payment

SEARCH INSURANCE PLAN

User should be able to search an Insurance Plan by PlanNo.

MODIFY INSURANCE PLAN

Search (By PlanNo.) an Insurance Plan and modify its details. System should show existing data/info of a Plan and support modify one, more or all info.

REMOVE INSURANCE PLAN

Search (by PlanNo.) a Plan and remove the Plan. System should ask for confirmation and on confirmation the data will be removed.

INSURANCE PLAN SUMMARY (VIEW)

System should show (display) Insurance Plan list in a tabular format (one row for each Plan, and columns for Plan details). It is not required to

show all the details of a Plan in a table; only important info like – PlanNo, Name, Description, etc. should be displayed.

2) **Customer**

SEARCH INSURANCE PLAN

User should be able to search an Insurance Plan by PlanNo.

MODIFY INSURANCE PLAN

Search (By PlanNo.) a Plan and view its details.

APPLY INSURANCE ONLINE

User should be able to apply for any policy online.

INSURANCE PLAN SUMMARY (VIEW)

System should show (display) Plan list in a tabular format (one row for each Plan, and columns for Plan details). It is not required to show all the details of a Plan in a table; only important info like – PlanNo., Name, Description, should be displayed.

Constrains

- Proper validation is required
- System must show appropriate massage on all activity (whether activity is successful or failure)

- User must have proper menu to select the activity (create, modify, search, view, remove) that user want to perform.
- For “Insurance Plan” Entity:
    - PlanNo.: Numeric. Should be 7 characters long. With first 2 characters upper case letters followed by hyphen and then 4 digits
    - Name : String
    - Description: String
    - Death Benefit: String
    - Maturity Benefit: String
    - Participation In Profits: String
    - PlanParameters: Numeric
- For “PlanParameters” Entity:
    - Age of Entry: Numeric. Between 18 - 50
    - Premium Paying Mode: String. (Yearly, HalfYearly, Quarterly, Monthly)
    - Policy Term: Numeric. Between 18 - 35
    - Basic Sum Assured : String
    - Policy Revival: String

- Premium Mode Rebate: String
- Loan: String
- Surrender: String

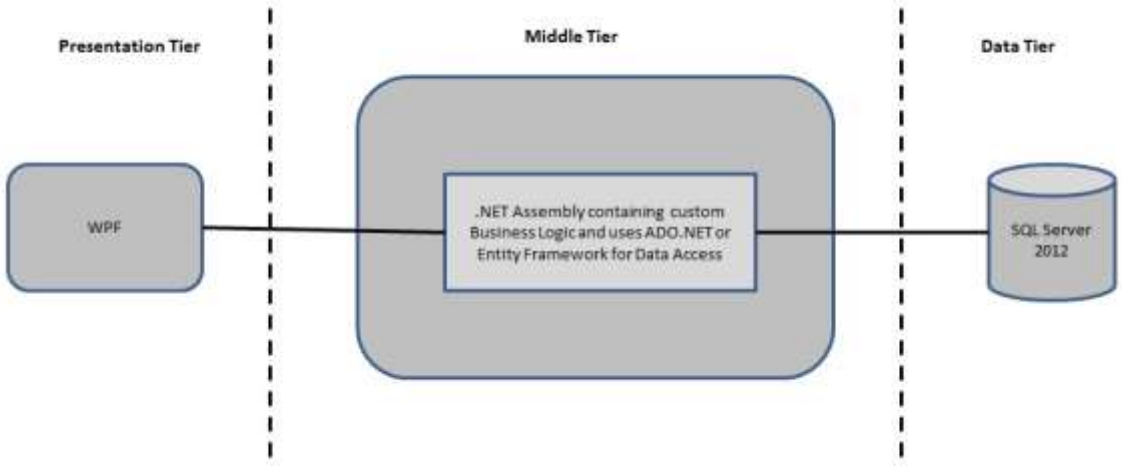
FUNCTIONAL COMPONENTS OF THE PROJECT

Application Architecture:

Distributed web applications traditionally to be designed and built across three logical tiers:

- Database Access Layer (DAL)
- Business Logic Layer (BLL)
- Presentation Layer

The DAL refers to the database itself, the stored procedures, and the component that provides an interface to the database. The BLL refers to the component that encapsulates all the business logic of the application. And, the Presentation layer refers to the web application pages.



Design guidelines

- All the exceptions/errors to be captured and user friendly message to be displayed on the CommonError page.
- Data access layer of 3-tier use Entity Framework data access using SQL stored procedures - All the database interaction would be performed using Data Access Component.

TECHNOLOGY USED:

- *Presentation Layer*
    1. *Console Application, WPF, ASP.NET MVC 5*
- *Business Layer*
    1. *Business Logic Components and Services :-*
        a. C# 5.0
- *Database Layer*
    1. *Databases:-*
        a. SQL Server 2012

IMPLEMENTATION

SUMMARY OF THE FUNCTIONALITY TO BE BUILT:

The participants need to develop the Customer Management System by building the functionality incrementally in each of the course modules of .NET LOT.

Sr. No	Course	Duration (in PDs)	Functionality to be built
1	MS SQL Server 2012	4	Creating relevant database tables and stored procedures
2	NET Framework 4.6 + C# 7.0 + Introduction to WPF	10	Developing Business components (C# classes)
3	ADO.NET with LINQ and Entity Framework	4	Creating data model and data context and using LINQ to entities
4	ASP.NET MVC 5	4	Incorporating advanced UI functionality with ASP.NET MVC 5
5	Mini Project Presentation	1	The Mini Project Presentation day

Note: Saturday half day will be devoted for Mini project

GUIDELINES ON THE FUNCTIONALITY TO BE BUILT:

The functionality and components to be built in each of the course modules of .NET LOT is as follows:

1. Course: SQL Server 2012

This section describes some of the basic steps involved in designing and creation of the database for the application.

Create Data Model - identify the different tables and fields that we will need, which would later be used for building the rest of the application.

Database Schema - Taking these objects, we can easily identify our main tables in the database.

a. Create the following database tables with following fields: [make your assumptions in case you require few more fields]

Table Name: InsurancePlan		
Field Name	**Constraint**	**Data Type**
PlanNo	Primary Key (System Generated)	Int
Name	Not Null	varchar
Description	Not Null	varchar
DeathBenefit	Not Null	varchar
MaturityBenifit	Not Null	varchar
ParticipationInProfits	Not Null	varchar
PlanParameters	Foreign Key (References Id column from Table - PlanParameterDetails)	int

Table Name: PlanParameterDetails		
Field Name	**Constraint**	**Data Type**
PlanNo	Primary Key (System Generated)	int

AgeOfEntry	Check(Between 18-50)	int
PremiumPayingMode	Not Null	varchar
PolicyTerm	Check(Between 18-35)	int
BasicSumAssured	Not Null	int
PolicyRevival	Not Null	varchar
PremiumModeRebate	Not Null	varchar
Loan	Not Null	varchar
Surrender	Not Null	varchar

